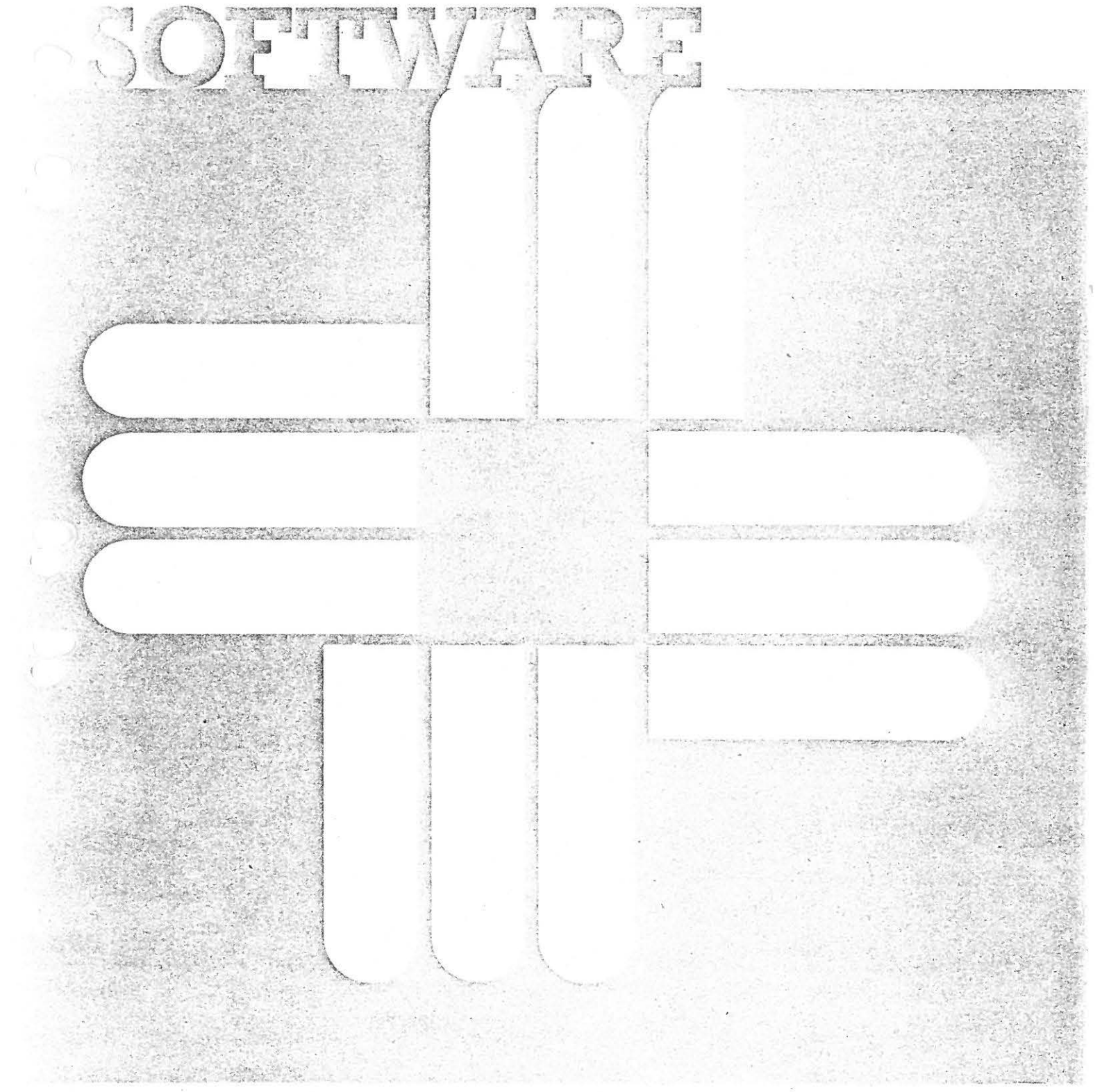
MODCOMP
MODULAR COMPUTER SYSTEMS, INC.

MAX IV System Guide

Basic I/O System



MANUAL HISTORY

Manual Order Number: 213-804005-I01

Title: MAX IV SYSTEM GUIDE, BASIC I/O SYSTEM

Software Part Numbers: 610304-000 I.1; 8240I

Revision Level	Date Issued	Description
H00	03/83	Initial Issue (H.0). (This manual replaces chapters 2 and 103 of the MAX IV INPUT/OUTPUT Reference Manual, 210-610501-000 and chapters 5 and 6 of the MAX IV GENERAL OPERATING SYSTEM Technical Manual, 220-610304-000.)
H01	11/83	Revision (H.1).
I00	09/84	Reissue (I.0).
I01	07/85	Revision (I.1). Corrections made to description of UFT status bits.

Contents subject to change without notice.

PREFACE

Audience

The information in this manual is directed to the applications programmer or developer with a good knowledge of MODCOMP Assembler language and familiarity with MAX IV SYSGEN and REX Services.

Subject

This manual contains information about the functions of the MAX IV Basic Input/Output System. Symbiont tasks and writing handlers is also presented.

Those users familiar with previous issues of MAX IV manuals may notice that the information contained in the MAX IV INPUT/OUTPUT, Reference Manual, 210-610501-000 has been reorganized to provide easier accessibility.

The information previously contained in the chapters listed below is now available as this MAX IV BASIC INPUT/OUTPUT SYSTEM, System Guide Manual.

Chapter 2 MAX IV BASIC I/O SYSTEM

Chapter 103 MAX IV SKELETON SYMBIONT TASK

Chapters from the MAX IV GENERAL OPERATING SYSTEM, Technical Manual, 220-610304-000 have also been included.

Chapter 5 MAX IV SYMBIONT TASKS Theory-of-Operation

Chapter 6 MAX IV WRITING OF HANDLERS Theory-of-Operation

Product Support

MAX IV Software System, Model 8240I.

Related Publications

Refer to the following manuals for additional information. When ordering manuals, use the manual order number listed below. The most current revision level (REV) will be shipped.

Manual Order Number	Manual Title
213-804001-REV	MAX IV GENERAL OPERATING SYSTEM System Guide (SGM)
213-804002-REV	MAX IV SYSGEN System Guide (SGM)
213-804003-REV	MAX IV EXECUTIVE (REX) SERVICES System Guide (SGM)
213-804004-REV	MAX IV OPERATOR COMMUNICATIONS System Guide (SGM)
213-804006-REV	MAX IV UNIT RECORD DEVICE HANDLERS System Guide (SGM)

213-804007-REV	MAX IV DATA STORAGE DEVICE HANDLERS System Guide (SGM)
213-804008-REV	MAX IV COMMUNICATIONS HANDLERS System Guide (SGM)
215-804004-REV	MAX IV UTILITY LIBRARY Library Reference Manual (LBM)
215-804005-REV	MAX IV EXECUTIVE FUNCTIONS AND PROCESS I/O Library Reference Manual (LBM)
211-829003-REV	MODACS III MODULAR DATA ACQUISITION AND CONTROL SUBSYSTEM Programmer's Reference Manual (PRM)
211-836001-REV	MODACS II MODULAR DATA ACQUISITION AND CONTROL SUBSYSTEM Programmer's Reference Manual (PRM)
211-804011-REV	MAX IV AND MAX 32 TASK OVERLAY CATALOGER Programmer's Reference Manual (PRM)

Special Symbols and Notations
A revision bar (|) located in the margin of the page in the text indicates a change to the manual. This change generally represents a technical change to the product due to product revision. A revision bar is also entered in the Table of Contents to flag the general location of changes in the text.

MODCOMP Product Training
MODCOMP's Education Services provides training courses on many hardware and software products at our Training Center in Ft. Lauderdale, Florida. On-site courses at customer sites can also be arranged. For more information about the courses offered by Education Services, contact the MODCOMP Training Registrar.

MAX IV REVISION I.Ø SUMMARY

MAX IV Revision I.Ø necessitated no changes to the information contained in this manual.

MAX IV REVISION H.Ø SUMMARY

TYPE Ø DISCS

Type Ø discs are those that support alternate track processing as well as dynamic determination of disc characteristics. Type Ø disc packs must be prepared by the Alternate Track Initializer Program.

When a pack is prepared for alternate track processing a number of cylinders are reserved for a spare track pool. The usable cylinders are the physical cylinders less the number of cylinders set aside for the spare track pool. The number of usable cylinders must be identified to the Operating System at SYSGEN to allow run-time compatibility. This is done with the LOGTRAN statement. The values specified for Parameter 4 on the LOGTRAN statements reflect the number of usable tracks seen by the Operating System for a logical transport. If multiple logical transports are defined for a physical transport, the total number of tracks specified in Parameter 4 (LOGTRAN statement) must equal the number of usable tracks on that physical transport.

PHYSICAL CYLINDERS - the total number of cylinders on the physical media.

USABLE CYLINDERS - the physical CYLINDERS less the number of cylinders set aside for the spare track pool.

USABLE TRACKS - the number of tracks on the disc as seen by the Operating System. This number is the value to be used in the LOGTRANSPORT statement at SYSGEN. USABLE TRACKS = USABLE CYLINDERS * HEADS

UNFORMATTED CAPACITY	SECTOR/ TRACK	HEADS	USABLE CYLINDERS	PHYSICAL CYLINDERS	USABLE TRACKS	PHYSICAL TRACKS
12MB*	64	2	305	320	610	640
16MB@	64	4	200	206	800	823
24MB*	64	4	305	320	1220	1280
32MB@	64	2	808	823	1616	1646
40MB	64	5	400	411	2000	2055
80MB+	64	5	805	823	4025	4115
150MB	64	19	405	411	7695	7809
160MB*	64	10	812	823	8120	8230
300MB	64	19	809	815	15371	15485
300MB	64	19	817	823	15523	15637

* Fixed Media
+ Fixed or Removable Media
@ Mixed Media - Fixed and Removable Media

Table S-1. CLASSIC Disc Drive Recommended Parameters for Type Ø

Table S-1 reflects that approximately 5 percent of the total disc capacity has been dedicated for alternate track allocation. This is a recommended amount. (See NOTE below.) The number of usable cylinders is determined when the pack is prepared by the Alternate Track Initializer Program.

NOTE: The total number of tracks allocated for alternate tracking must not exceed the capacity of the controller's buffer. The controller's buffer space is 120 tracks per disc transport.

The reader is referred to the following manuals:

MAX IV SYSGEN, System Guide Manual

Alternate Track Processor, Programmer's Reference Manual

MODCOMP Stand-Alone Support Software, Programmer's Reference Manual
ALTERNATE TRACK INITIALIZER

MAX IV Data Storage Device Handlers, System Guide Manual

LOGICAL TRANSPORTS

MAX IV can treat the fixed and removable platters of mixed media discs as separate discs. This allows the system to believe that there is more than one disc transport per physical disc drive.

MAX IV Logical Transport Support (LTS) gives the user the ability to configure a mixed media drive as separate transports. This feature allows a combination of BIOS and File Manager support on the same physical disc transport. The reader is referred to the following manuals:

MAX IV DATA STORAGE DEVICE HANDLERS, System Guide Manual
Chapter 3 MAX IV CLASSIC DISC DEVICE HANDLER
Chapter 4 MAX IV FIXED AND MOVING-HEAD DISC HANDLERS
Chapter 5 MAX IV LARGE CAPACITY MOVING-HEAD DISC HANDLERS
Chapter 6 MAX IV BULK MEMORY HANDLER

MAX IV SYSGEN, System Guide Manual

DISCTRANSPORT SYSGEN STATEMENT

The DISCTRANSPORT SYSGEN macro is not supported in MAX IV Revision H.0. Systems that use this statement should substitute an appropriate FMTRANSPORT statement for each occurrence of the DISCTRANSPORT statement.

NOTE: The order of parameters is not the same for the two statements. The reader is referred to the MAX IV SYSGEN, System Guide Manual.

STREAMER TAPE DRIVES

MODCOMP has updated its magnetic tape controllers to support streamer tape drives. These drives allow for the continuous transferal of data without required delays for inter-record gaps.

Streamer tape support is available to the user as an option in the Extended UFT Option Word. It is the user's responsibility to access the streamer tape drive in a manner suitable for such a device.

SEVEN TRACK MAG TAPE

Seven-track mag tape drives that use 556 bpi density or even parity are not supported under MAX IV Revision H.Ø. The bits in the LDT option word that were used to support these features are not recognized by Revision H.Ø. This change was necessary because of modifications made to the magnetic tape controller hardware to support streamer tape drives.

MAX IV Asynchronous Terminal Handlers - AS.HAN, AS7.HA

TERMINAL LISTING CONTROL

Terminal Listing Control (TLC) allows the terminal user to hold or resume output to the terminal screen. This capability is selected at SYSGEN in the option parameter of the COMDEVICE statement.

TLC is supported for both full and half-duplex terminal users in either Data Interrupt (DI) or Direct Memory Interface (DMI) channel environment.

When TLC is enabled, the system recognizes default listing control characters. The LISTCONTROL SYSGEN statement may be used to define special listing control characters for the listing stop or listing resume functions. An automatic listing resume time interval may also be specified on the LISTCONTROL statement.

BUFFERED DEVICE SUPPORT

Buffered Device Support (BDS) provides interface to a buffered device in two ways. One involves RS232C signal monitoring and the other relies on input data patterns. BDS is selected at SYSGEN by setting a bit in the option parameter of the COMDEVICE statement.

Buffered Device Support is provided for both half-duplex and full-duplex modes of operation. When a buffered device indicates buffer full by sending an X/OFF character to stop output transmission, the associated handler will attempt to perform the function as quickly as possible. X-ON - X/OFF character support is handled in a manner similar to that used by Terminal Listing Control. The RS232C signal support technique is primarily controlled from the hardware channel level.

MAX IV Asynchronous Handlers - ASHAN, AS7.HA
MAX IV Bisynchronous Handlers - BI.HAN, BI7.HA, SY7.HA
MAX IV SDLC/HDLC Handlers - DLC.HA

INTERNAL WRAPROUND SUPPORT

Internal Wraparound (IW) support allows the output subchannel of a duplex channel to transmit to the input subchannel of the same duplex channel. This mode of communication is useful when diagnosing system problems or when a need exists to test out software without having to worry about the external channel line characteristics.

A privileged state is required in order to use this feature. The user task must also have exclusive use of the duplex channel pair to use this feature. Internal Wraparound is enabled or disabled by setting bits in the Extended UFT Option Word (XUFOPT) and using the REX HOME service. Only REX HOME services issued to the output channel (even numbered channel) affect IW states.

RING DETECT SUPPORT

Ring Detect Support provides the user with an auto answer mechanism that may be tailored to suit most system environments. Auto task activation (connect) and task deestablish/kill (disconnect) capabilities may be optionally selected. Hooks are available within the Terminal Monitor Program (TMP) that enable the user to perform special functions on detection of RING, BREAK, and HANG UP events.

Ring Detect Support gives user a mechanism that allows remote connection to a MODCOMP communications line. Support includes asynchronous, byte synchronous and bit synchronous line interfaces.

The new LINEMONITOR SYSGEN statement is used to configure communication channels as either Break character/Key supported and/or Ring supported. The old BREAK SYSGEN statement generates a SYSGEN error under MAX IV, Revision H.Ø. The BREAK statement's optional password argument can be specified with the new OCPASSWORD SYSGEN statement.

WARNING

AS OF MAX IV REVISION H.Ø, THE FOLLOWING USER INTERFACE CHANGES APPLY.

AS.HAN and AS7.HA - REX HOME function no longer performs the same as REX REWIND. REX HOME is used to enable/disable Internal Wraparound.

BI.HAN, BI7.HA and SY7.HA - Previous to MAX IV, Revision H.Ø, the REX HOME service would reset DTR in the duplex channel interface. As of Revision H.Ø, the REX HOME service has the capability of resetting DTR and performing IW features also. Concerned users should refer to Chapter 3, MAX IV BISYNCHRONOUS COMMUNICATIONS HANDLERS.

SL.HAN - As of MAX IV, Revision H.Ø, the REX REWIND service is a null operation. Users are already required to be privileged.

MAX IV SDLC/HDLC Frame Handler - DLC.HA

DI ENHANCEMENT

Interrupt and Transfer (DI) and half-duplex capabilities are available in the MAX IV SDLC/HDLC Frame Handler under MAX IV, Revision H.Ø. To use the handler in DI mode, the user needs to specify class 'H' on the COMCONTROLLER SYSGEN statement for the SDLC/HDLC channel being configured. To use the handler in half-duplex mode, only one COMCONTROLLER and COMDEVICE statement should be specified for each channel pair.

MAX IV REVISION H.1 SUMMARY

Revision H.1 release of this manual contains the revised HOLD command text.

UNFORMATTED CAPACITY	SECTOR/ TRACK	HEADS	USABLE CYLINDERS	PHYSICAL CYLINDERS	USABLE TRACKS	PHYSICAL TRACKS
.75MB	16	2	80	80	160	160
12MB*	64	2	305	320	610	640
16MB@	64	4	200	206	800	823
24MB*	64	4	305	320	1220	1280
27MB*	32	8	300	310	2400	2480
32MB@	64	2	808	823	1616	1646
40MB	64	5	400	411	2000	2055
80MB+	64	5	805	823	4025	4115
150MB	64	19	405	411	7695	7809
160MB*	64	10	812	823	8120	8230
300MB	64	19	809	815	15371	15485
300MB	64	19	817	823	15523	15637

* Fixed Media
+ Fixed or Removable Media
@ Mixed Media - Fixed and Removable Media

Table S-2. CLASSIC Disc Drive Recommended Parameters for Type 0

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1
BASIC INPUT/OUTPUT SYSTEM (BIOS)

A MAX IV programmer requests I/O operations using a series of Request for Executive (REX) service calls (for example, READ, WRITE, REWIND). Each of these service routines uses a common central subroutine called the Basic I/O System (BIOS). For all tasks, BIOS schedules all I/O operations to the controllers of real devices or to symbiont tasks for imaginary devices. The Basic I/O System is re-entrant and may be called concurrently by many tasks.

1.1 LOGICAL FILES

The programmer does not actually address an I/O device directly when requesting an I/O operation through BIOS. Instead, the programmer directs all I/O operations to a logical file, with a name that is arbitrarily chosen and is one of the arguments of all BIOS service calls. BIOS requires that this logical file be assigned to a target device (real or imaginary) before useful work can be done.

The advantage of the logical file intermediary is that programs can be written that can operate with any similar I/O device (device independent). The program is more adaptable, allowing it to use different actual devices at any time. This is essential in a system where device resources are assigned to programs as they are available. Assignment of the logical file to its target device can be accomplished at system generation time, at task cataloging time, or at run-time.

1.2 REAL DEVICE HANDLERS

Each device controller in the system has a resident controller Physical Device Table (PDT) which contains information about the physical characteristics and status of the device controller and one or more Transport Control Tables (TCT) that also contain the particular transport's slough queue.

The PDT in which the Active Node Queue (PDQ) head is located, is also used for any re-entrant temporary storage necessary during the life of a particular operation. For example, during READ or WRITE operations, it contains pointers and counters to the current byte or word position in the program's data buffer. Therefore, each data interrupt can communicate information to subsequent data interrupts and to the service interrupt when the operation is terminated.

1.2.1 INTERRUPT PRIORITY LEVELS

Most I/O device controllers have their own unique hardware group/unit address and unique interrupt transfer locations. Because of the unique interrupt trap address, device identification is automatic and is thus accomplished with low overhead. The

interrupts from all I/O devices are combined on two unique priority levels. The two levels are the Service Interrupt (SI) level and the Data Interrupt (DI) level. The DI is the higher level and is associated with data transfer into and out of memory. The SI indicates some status condition that may need special processing, for example, an End-of-Block condition. Within each of these levels, seventeen unique device sub-priorities are defined by hardware.

The data interrupt level (or party line) combines all data interrupts of all I/O devices so that one data interrupt cannot interrupt another data interrupt. This feature allows common data handling software elements to be shared with all devices without the usual overhead of re-entrant routines (these common elements are serially reusable, automatically). Because of the party line nature of the DI level, all DMP devices can share commmon software elements at this level because they never interrupt each other.

The service interrupt level (or party line) combines all the service interrupts of all I/O devices but is lower in priority than the data interrupt party line. One service interrupt cannot interrupt another service interrupt, thus allowing common device service software elements to be shared with all devices as in the data interrupt case above. Because data interrupts occur at greater frequency, they are higher in priority than service interrupts and can interrupt service interrupt routines.

The memory savings are significant with this double party line interrupt scheme, without the usual data rate restrictions imposed by machines that have a single party line for both data and service interrupts and without the usual cost penalty imposed by machines that require two unique priority interrupt levels for each device.

1.2.2 SHARED HANDLER

Additional memory is saved when there are multiple devices of the same type. In this case, not only can certain common task, data, and service elements be shared, but the entire resident handler for a particular type of device can be shared by several controllers. Only the controller Physical Device Table (PDT) and a pair of entry drivers need be duplicated for such devices.

The system-generation procedure allows the user to configure any combination of the I/O handlers and to share a single copy where applicable. User-coded handlers for non-standard devices may be added to the operating system.

1.2.3 PROCESSING LEVELS

Each device handler is organized into three levels of processing. They are:

- Task Level
- Service Interrupt Level
- Data Interrupt Level

The Task Level portion of the handler provides the housekeeping requirements that are necessary for device processing and the functions necessary to ensure device independence. Any functions normally considered as high overhead processes (such as converting a file position index into actual disc address coordinates) are best implemeted here as these functions occur at the priority of the calling task.

The controller's service interrupt routine initiates transfers and processes termination of all operations. Once many operations are queued for a particular controller, this interrupt routine processes all the operations without any assistance from the original calling task. The controller's data interrupt or Direct Memory Processor (DMP) transmits the actual data (word or byte at a time) required during an operation between service interrupts.

1.3 IMAGINARY DEVICE HANDLERS

The MAX IV I/O System recognizes special "imaginary" devices that may be defined at system generation time. These devices have all the characteristics of a real interrupt-driven device. They are capable of being assigned to any file and having I/O operations queued for them. Whenever these operations are queued, another task, called a SYMBIONT, is "activated" instead of a group/unit channel and interrupt being triggered. The symbiont task initiates the operations and handles the completion of the operations just as a real device handler would do.

Normally, the symbiont operates at a task priority level higher than any task that may call it. A symbiont may be a resident task or may be a prescheduled non-resident task.

Symbionts can be used to simulate devices, to impose intermediate disc buffering (spool), or to impose special formatting, protocol, or data conversions on existing interrupt-driven devices. These techniques can be implemented in such a way that they are transparent to the task that uses them. Several types of standard symbionts are described. in separate chapters of the MAX IV Unit Record Device Handlers, System Guide Manual.

A calling task READS from, and WRITES to, the imaginary device as if it were a hardware device. Each READ, WRITE, or other I/O operation is queued to the symbiont and causes an I/O node to be transferred into the symbiont. During WRITE operations, the contents of the calling tasks write buffer are transferred to a buffer within the symbiont. For READ operations, the contents of the symbiont buffer are moved to the read buffer of the calling task.

1.4 ACTIVE NODE QUEUE

The MAX IV I/O system allows unlimited levels of queuing to common or separate devices. MAX IV queues operations according to calling task priority level and then chronologically for like levels.

The MAX IV queuing capability allows the programmer to continue execution even though the file/device requested may be busy with other operations already requested by the program or another task. A task can even queue many operations to the same device without execution being suspended.

Queuing is accomplished by linking all operations into a linked list of operations. Each device controller or symbiont task has its own queue--since this is where all "bottlenecks" occur. Some devices have their own controller or symbiont task (thus, their own queue) while other devices share a common controller or symbiont task (and thus, share a queue).

An operation in a queue is characterized as a table (single dimension array) called a node. All pertinent information that describes the operation is copied into the node. These I/O nodes (or operations) are then strung into the controller's queue (in the Physical Device Table) in an order appropriate to the task priority level at which the operations were called. When the same task priority level queues more than one operation, the nodes are strung in chronological order for that level.

The first node in a queue is the currently initiated operation. All operations that arrive after the first node is initiated are queued. Exceptions are made for certain devices such as teletypewriters. For these devices the currently initiated operation is interrupted in favor of higher priority operations.

When an operation is completed and is error-free, the controller's service interrupt routine (or symbiont task) removes the node from its queue and lets the task know that the operation is completed. The SI routine resets a bit in the calling program's UFT, resets the task's I/O HOLD bit if Wait mode was used for the operation and generates a TASKMASTER interrupt. The node is returned to the available list so that it may be used for another operation. If the service interrupt routine (or symbiont task) finds another node in the queue, it initiates the indicated operation and then exits until the operation is completed or abnormally terminated.

1.5 I/O DEVICE HANDLER MODES

1.5.1 I/O OPERATION MODES

The I/O device handlers that make up the Basic Input/Output System have "device-independent" modes of operation which attempt to minimize or abolish the differences between several similar but somewhat different devices.

If this device independence is not required, the programmer has the freedom to use other "device-dependent" modes of operation. Device-dependent programming allows the user to deal directly with the peculiarities or advantages of a particular device. The device-dependent features are described with each handler.

1.5.2 DATA TRANSFER MODES

MAX IV BIOS performs data transfers in on of three modes:

- Data Interrupt (DI)
- Direct Memory Processor (DMP)
- Direct Memory Interface (DMI)

The data transfer mode is dependent upon the data transfer rate of the target device.

Devices that have data rates too high for data interrupts on a word or byte basis use a Direct Memory Processor (DMP). These devices transfer blocks of data in and out of memory automatically without program control. In such devices, the data interrupt is used as an end-of-block interrupt.

BIOS handlers that perform I/O in the DI mode are referred to as non-DMP handlers.

1.5.2.1 DMI Mode

A third mode of data transfer is the DMI mode. DMI transfer requires special hardware -- a Communications Processor (CP) and a Universal Communications Subsystem (Model 1907) or a CP and an Integrated Communications Subsystem (Model 1908).

The Communications Processor extends the hardware instruction set and provides an external multiplexed data path between main memory and the Universal or Integrated Communications Subsystems.

The extended instruction set is optimized for handling message oriented data. Instruction capabilities include high-speed byte string MOVE operations with the ability to perform character translations, editing functions, and computation of block-check characters. The implementation of the instructions is in the hardware enabling the normally time-consuming functions of message formatting to be performed at register-to-register speeds.

The external multiplexed Direct Memory Interface (DMI) provides a data path directly to and from main memory. This data path is independent of the Central Processing Unit (CPU) and provides for message mode transfers up to 256 full-duplex communications lines. Each communication line is controlled by its own discreet special character algorithms, enabling character detection to be performed in hardware. The individual algorithms are dynamically definable by software that provides the flexibility to accommodate changing environments.

Communications data throughput is increased by an order of magnitude by the combination of hardware message processing instructions and elimination of the high-overhead software for character I/O processing.

The programmable characteristics of the editing functions and special-character algorithms provide the flexibility to accommodate new message processing and line protocols.

Refer to the following chapters in the MAX IV Communications Handlers, System Guide Manual for more information on DMI mode:

Chapter 2 The MAX IV Asynchronous Communications Handlers (AS7.HA)

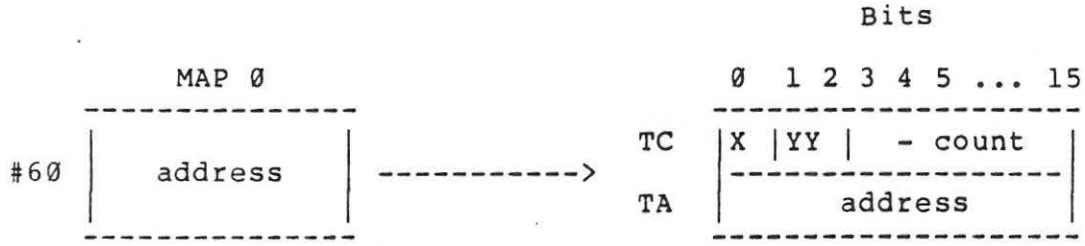
Chapter 3 The MAX IV Bisynchronous Communications Handlers (BI7.HA and SY7.HA)

1.5.2.2 Data Chaining

The MAX IV I/O system supports data chaining for most DMP devices. Only privileged tasks may use data chaining. This feature provides the programmer with the capability to output one physical record from non-contiguous memory locations and also to input one physical record into non-contiguous memory locations. This is sometimes called scatter-read and gather-write. Data chaining is accomplished by a list of transfer counts and addresses (called the chain list), which is sequentially accessed by the DMP during a data transfer.

DMP operations in the CLASSIC 7820's - 7870's series computers require that several words in MAP Ø be set up prior to issuing the I/O commands to the device controller. To get into MAP Ø space requires that the task be privileged. The reason for entering MAP Ø is to pass information to the DMP I/O processor hardware. Part of this information is passed as a pair of words in MAP Ø known as the Transfer Count (TC) and the Transfer Address (TA). Normally there is a dedicated TC/TA pair of words per DMP channel. This initial TC/TA pair can point to other nondedicated TC/TA pairs to be serviced in one continuing DMP operation, known as data chaining. Regardless if data chaining is used or not, the first pair is always found in MAP Ø. This first pair is pointed to by Word #6Ø in MAP Ø. This means that the first pair can be located anywhere in MAP Ø space. The operating system (BIOS) uses specific locations defined by convention and convenience.

Words #62 and #63 are the TC/TA for DMP channel 1, #64 and #65 are for DMP channel 2 ... #7E and #7F are for DMP channel #F. To find the TC/TAs for the optional hardware channels #1Ø-#3F, look to location #61, that has the TC/TA address for channel Ø in it, followed by the #1Ø TC/TA pair. Higher channel numbers follow in consecutive word pairs. The format for the TC/TA is shown in Figure 1-1.



X is: Ø = Chain DMP operation.
1 = Last TC/TA in a list.

YY is: ØØ = New chain list is in MAP Ø.
Treat TA as an address to this list.
Ø1 = New chain list is in the tasks map.
Treat the TA as an address in the users program.

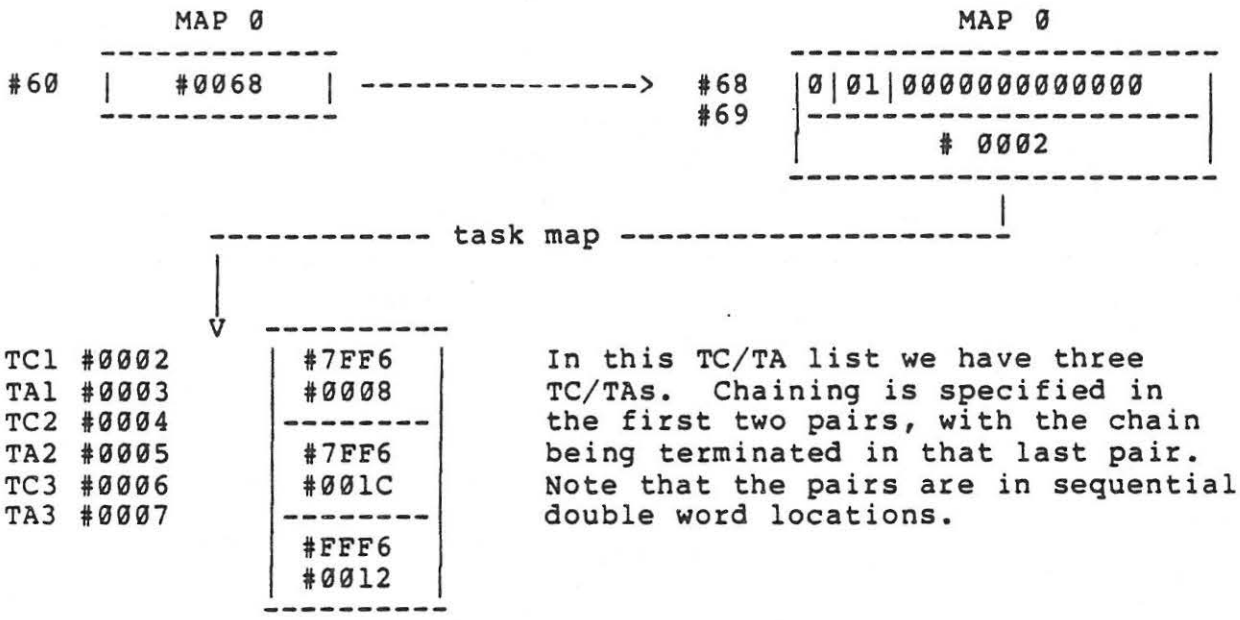
1Ø
or = Continue in this list. If data chaining,
11 the next pair follows this TC/TA. The TA is a valid data address. The YY field is a part of the negative word count (16 kw/block).

Figure 1-1. TC/TA Format

The TC can represent the number of words to be transferred in this block, or it can be used to transmit information to the I/O processor concerning where the next TC/TA pair can be found. The latter type is classified as a control TC. With this type, the TC/TA pair is not used to perform I/O, but is used to pass commands to the I/O processor. These commands (see the YY field, ØØ and Ø1 above) must point the I/O processor to a valid TC/TA pair.

The TA is used as a pointer to other TC/TAs (TC Command), or is used as a valid virtual starting buffer address for the data to be read into or written from. Regardless of where the TC/TAs come from, the map image to be used by the I/O processor to locate the data buffer must be passed to it by the DMPI instruction prior to the initiation of the data transfer.

The following data is an example of a DMP operation to a magnetic tape unit on device address and DMP channel 4. To prepare for an I/O operation, MAP Ø location #6Ø is set to point to location #68 (the TC/TA pair) with the appropriate TC/TA placed within that double word location. Refer to the following example.



The initial TC/TA in MAP Ø specifies that the TC/TAs to be used are going to be found in the tasks virtual space starting at Word 2. In this example, both the data and the TC/TAs are located in the users program. Regardless of where the TC/TAs come from, the map image to be used by the I/O processor to locate the data buffer must be passed to it by the DMPI instruction prior to the initiation of the data transfer.

Please be aware that the example that follows makes no attempt to check the device status, DPI, or operation completion states. It is suggested that this would be desirable in a user environment. In this example, the end of file and other positioning commands were performed through Job Control, or the Source Editor.

The chain list may be mapped into the same addressing space as the I/O buffers or it may reside in MAP Ø. This feature is implemented principally for a MAP Ø task that wishes to load information into the addressing space of another task with a chain list built in its own addressing space.

1.5.3 DATA FORMAT

Symbolic data is always transmitted to and from memory buffers in USASCII code. This is the adopted internal code used by MAX IV for all symbolic transfers. The code set, a maximum of 128 defined data and control characters, requires seven bits to define each character, but each character always resides in an 8-bit byte, with the most significant bit unused. Two such bytes reside in each 16-bit memory location.

MAX IV BIOS processes data in the following formats:

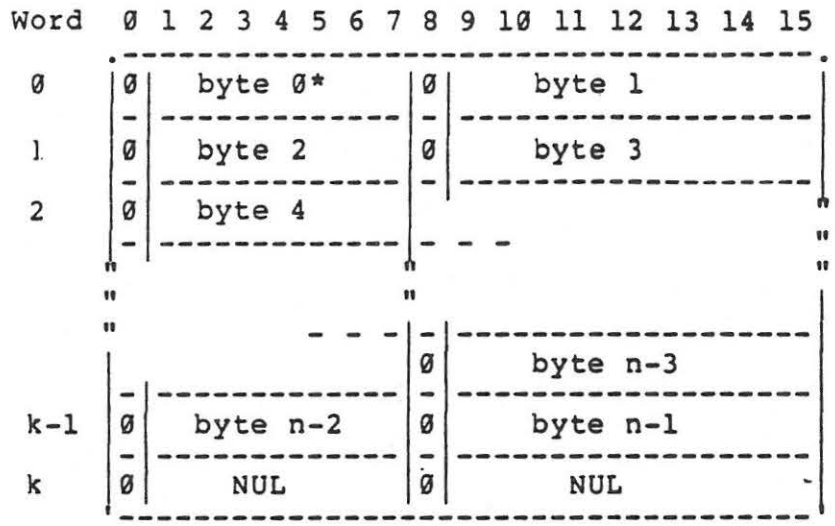
Standard ASCII
Non-Standard ASCII
Standard Binary
Non-Standard Binary

The specific handler chapters explain the difference in processing in each format.

1.5.3.1 Standard ASCII

Figure 1-2 illustrates a Standard ASCII (symbolic) record as it might appear in a programmer's buffer before a WRITE operation or after a READ operation. All I/O handlers that operate devices capable of symbolic output (printers and secondary storage devices) have ASCII modes that allow the programmer to transmit such unit records.

n-byte record where n is even.
(Word k is not transmitted if
not within bounds of buffer.)



n-byte record where n is odd.
(Word k is not transmitted if
not within buffer.)

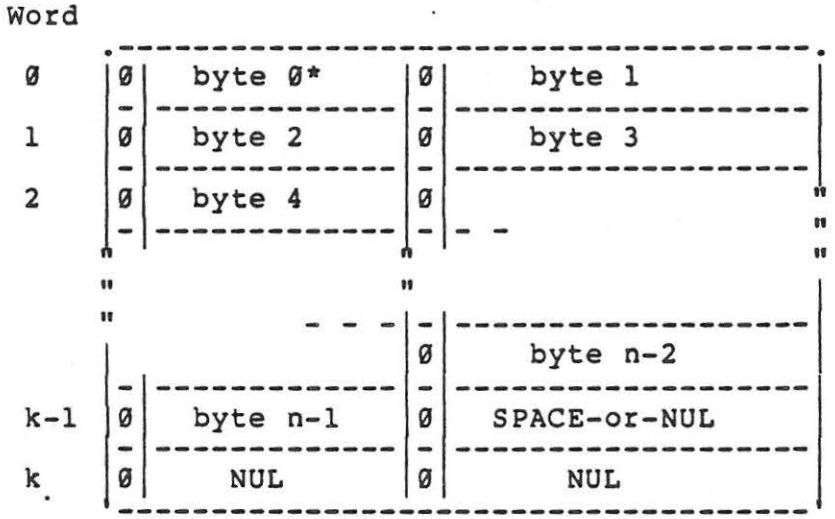


Figure 1-2. Variable Length Record in Memory - Standard ASCII

*NOTE: Byte 0 can never be ETX (#03) or BEL (#07) in the Standard ASCII mode since these bytes identify the beginning of a Standard Binary record.

If convenient, two Null (NUL=00000000) bytes, residing on a word boundary, may be imbedded in the programmer's buffer to mark the end of useful information. Most non-DMP devices terminate a WRITE when they encounter such NUL bytes regardless of the buffer size specified in the programmer's WRITE arguments.

Most non-DMP handlers also put such a NUL byte sequence in the programmer's buffer on READ operations to mark the end of variable length unit records. Such end-of-record markers are also convenient to syntax analyzer routines that process character strings in these I/O buffers in a sequential manner. (See the GET and COLLECT services described in the MAX IV Executive (REX) Services, System Guide Manual listed in the PREFACE).

1.5.3.2 Code Conversion

If a particular device uses an external hardware code set other than USASCII, the device handler usually converts the internal codes to desired code by either hardware (ROM table conversion) or software (indexed table look-up) means. The description of each handler indicates whether translation is performed for a particular device. In some cases, code conversion must be done by the programmer before/after the WRITE/READ operations are performed. In other cases, it may be easy to substitute a modified conversion table at system generation. It is also possible that a symbiont task can be written to perform code conversion when the original task addresses a particular symbiont device.

1.5.3.3 Carriage Control

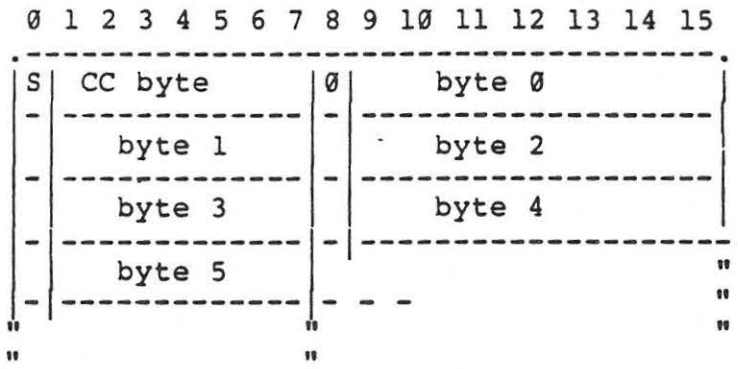
Symbolic data records that go to printing devices must contain control data that inform the device handler how to position the carriage for each WRITE operation. This carriage control data is usually transmitted in byte 0 of the programmer's data buffer. Devices that do not print usually ignore such data but transmit it as normal data to the device. Printing devices interpret this data and convert it into carriage positioning commands or special byte prefix sequences before the print data is transmitted. The carriage control data itself is not transmitted to a printing device. To defer the start of printed text to byte 2 (next word boundary) use bytes 0 and 1 for carriage control data. This is shown in Figure 1-3.

Bit 0 of Byte 0 determines whether the carriage control (CC) byte is a software carriage control character or hardware carriage control character. If Bit 0 is 0 the CC byte is interpreted by the handler for the carriage control command.

When Bit 0 is 1 it indicates hardware carriage control. The hardware code is device dependent.

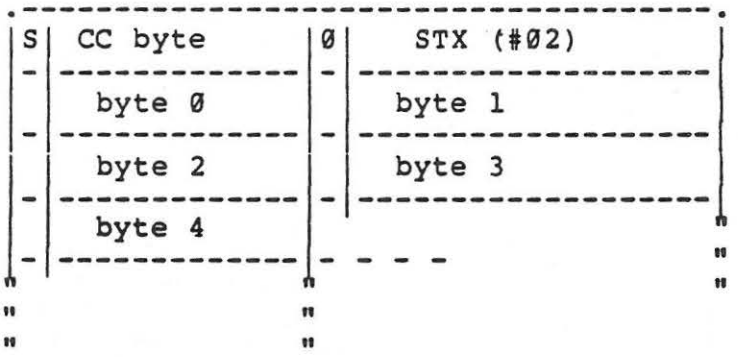
Some printing devices will skip to column "n" or upspace "n" lines before printing. Some devices do not support channel skipping operations.

SINGLE BYTE CARRIAGE CONTROL



FIRST PRINTED BYTE (byte 0) is in word 0.

DOUBLE BYTE CARRIAGE CONTROL



FIRST PRINTED BYTE (byte 0) is deferred to word 1.

S = 0 Software carriage control.

S = 1 Hardware carriage control.

Figure 1-3. Variable Length Standard ASCII Records to Printing Devices.

1.5.3.4 Binary Data Formats

Non-symbolic data is transmitted bit-for-bit from memory to device in most cases. If the target device is byte oriented, one 8-bit data byte in memory is transmitted to/from the 8-bit wide data path of the device. Two such bytes reside in each 16-bit location of the programmer's data buffer. The Standard Binary record format, as it would appear in a programmer's buffer, is shown in Figure 1-4.

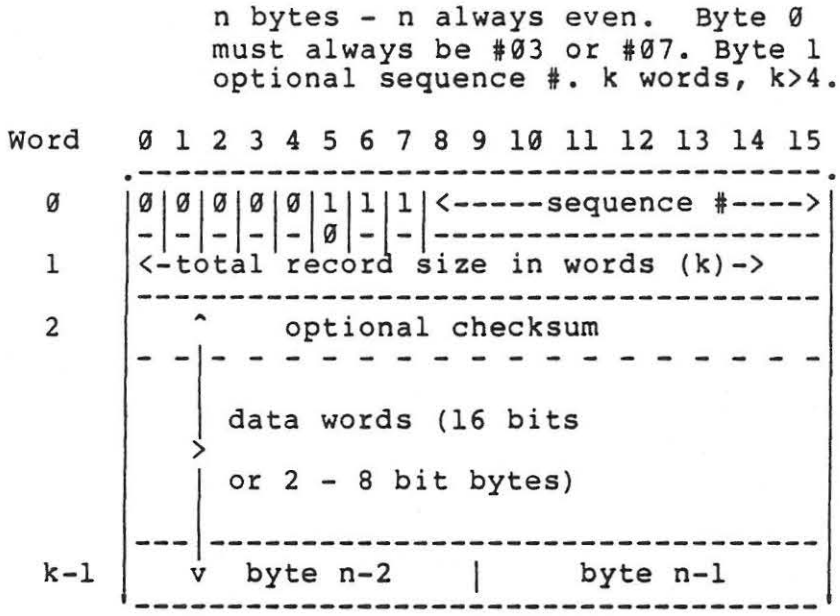


Figure 1-4. Variable Length Record in Memory - Standard Binary

The Standard Binary record format requires that Bytes Ø, 2, and 3 be reserved for control purposes so that such records can be identified as binary records and variable length binary records can be defined by control bytes within the unit record itself. If this format is restrictive most handlers have a Non-Standard Binary mode. However, most MODCOMP standard software uses the Standard Binary format for non-symbolic data or compressed symbolic data.

Device handlers interpret binary data in various ways. Secondary storage devices usually transmit it unmodified. Some devices encode/decode it into bit patterns appropriate to the device. Other devices, such as plotting devices, may even interpret such data as functional commands. The chapter on each individual handler describes how a particular device treats binary data.

1.5.4 I/O CONTROL MODES

The programmer may choose to address an I/O device in two modes when calling BIOS services:

- Wait mode: the service does not return control to the calling program until the requested operation is completed.
- Quick Return mode: the service returns control to the calling program as soon as the operation is queued.

The program may continue execution if there is work that it can do. It must eventually test for completion of the queued operation. If the operation is not ready when tested, the task has the choice of continuing to execute or giving up control of the CPU to lower priority tasks.

Some process I/O devices accept or take data without delay and thus ignore the concept of Wait and Quick Return modes.

1.5.5 ERROR RECOVERY MODES

One of two error recovery techniques may be selected by the programmer at the time a BIOS service is called:

- System Error Recovery
- User Error Recovery

The System Error Recovery mode frees the programmer of error testing at the completion of each device operation. Only events (end-of-file, et cetera) need be tested. Errors detected by the I/O handler cause appropriate device-dependent error-recovery procedures to be instituted by the handler. If these procedures fail, the device is placed offline and the error condition is reported to the operator. The operator may then decide what is appropriate for recovery.

The User Error Recovery mode places responsibility for error correction directly on the programmer. There are two modes of User Error Recovery. In one mode, the I/O system first attempts to recover from the error and if recovery fails, the error is returned to the calling task. At this point, the task may do whatever is necessary to recover (for example, device substitution, et cetera.)

In the other User Recovery mode, the I/O handler returns to the calling task as soon as an error is detected. When the operations are completed, the programmer must test for errors, then proceed to analyze the type and seriousness of the error, and then institute retry and/or device substitution techniques to a level appropriate to the application.

These error recovery modes are convenient since they allow some programs to run in a batch environment (when an operator is present) while higher priority real-time programs can run without operator assistance with each program instituting a level of error recovery appropriate to what it is doing. The System Error Recovery mode is useful for running non-batch tasks with an operator present or a series of tasks that are in a state of development.

1.6 AN I/O OPERATION

Figure 1-5 illustrates the interaction of four major operating system elements during the life of an I/O operation. It is simplified for clarity since it does not show the interaction and overlap between several different tasks, each making I/O calls. Time is not drawn to scale in the illustration. Data Interrupts or Direct Memory Processor (DMP) Cycles are not shown.

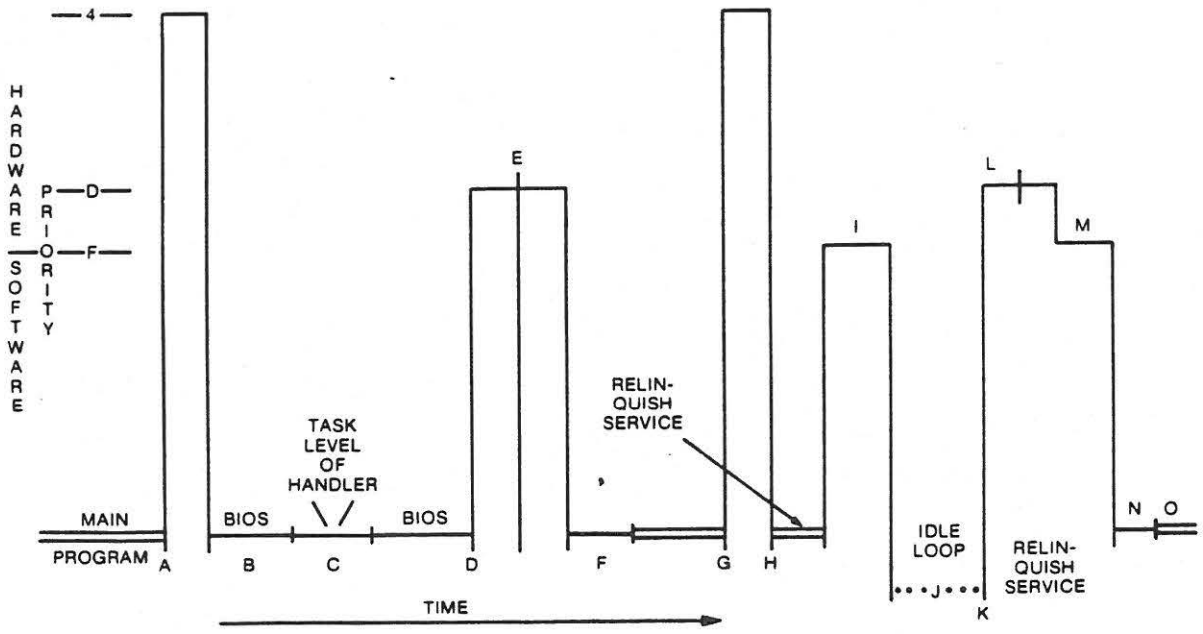


Figure 1-5. Major Events of a Task I/O Call.

Figure 1-5 can be explained as follows:

- A program issues an I/O call at (A).
- All call entries to BIOS are made using REX services that cause an immediate trap to the unimplemented instruction trap, followed by an entry to the required service at the level of the calling task (B).
- The central scheduling subroutine of BIOS resolves all arguments of the call, finds the device assigned to the logical file argument, and enters the task level portion of the device handler (C).
- The handler sets up all parameters necessary for the I/O operation and returns to BIOS. BIOS then proceeds to queue the operation to the device controller while locked at the service interrupt level (D).
- At (E), the BIOS subroutine finds this is the only operation in the controller queue and terminates the controller channel to force a real service interrupt entry as soon as it unlocks the level. This interrupt is immediate. The real interrupt subroutine of the controller's handler performs the device-dependent initiation functions to start the I/O operation.
- When it exits, control is returned to the calling task at (F). The main program can proceed concurrently with the scheduled I/O operation. Data transfers are by Direct Memory Processor (DMP) cycle stealing or data interrupts at device-dependent intervals (not shown).
- Eventually, the program is unable to continue executing until the device operation it has queued is finished. It then calls the RELINQUISH REX service at (G). This routine checks the necessary criteria in the I/O argument list to see if the operation is complete.
- If not, the service causes the task to relinquish time by triggering the Taskmaster (H) to possibly activate lower priority tasks since it can do no further useful work.
- The Taskmaster (I) finds no lower priority tasks active and thus transfers control to the "Idle Task" (J).
- At (K), the service interrupt that signals completion of the device operation occurs. It checks the device operation for errors and dequeues the operation node. It then triggers the Taskmaster so that it will inform the recipient task.

- If other operations had entered the controller I/O queue at this point, this routine would proceed to initiate the next operation (L). In this case, no such operation is found in the queue and the queue is empty. The service interrupt exits. Since the Taskmaster was triggered within the Service Interrupt (SI) routine, it immediately executes, following the exit of the higher hardware priority SI.

- The Taskmaster performs a scan of the CPUQ (M) and dispatches CPU control to the currently active highest priority task which, in this case, was the task that requested the completed operation.

- This task is still within the relinquish service. This service (N) again checks the criteria necessary to determine if the I/O operation was completed.

- Since these conditions are now met, the RELINQUISH service immediately returns control to the main program (O) which can now proceed with its execution.

NOTE: The I/O WAIT service is a more efficient means than the RELINQUISH service for a program to wait for an operation to complete. When entered, this service causes the calling task to be placed in an I/O hold until one or more specified operations have completed. This relieves the Taskmaster from the burden of entering a task in the RELINQUISH service only to have the task relinquish back to the taskmaster. Use of I/O wait improves total system performance as well as provides the improved ability of a task to wait for one or all of a series of I/O operations to complete.

1.7 REX I/O SERVICES

The REX services listed in Table 1-1 perform or queue physical I/O control and data transfer operations using logical files. These services are discussed individually in the MAX IV Executive (REX) Services, System Guide Manual listed in the PREFACE. All these services converge on the central scheduling routine of the BIOS.

Call Number	Call Name	Description	FORTRAN MAX IV Utility Library Subroutine	
/#Ø	READ	Read one physical record from an I/O device.	READ4	\ DATA
#1	WRITE	Write one physical record to an I/O device.	WRITE4	/ TRANSFER
/#2	REWIND	Rewind an I/O device.	REW4	\
#3	BKFILE	Backspace file on an I/O device.	BKFIL4	
#4	BKRECORD	Backspace record on an I/O device.	BKREC4	
#5	AVRECORD	Advance record on an I/O device.	AVREC4	
#6	AVFILE	Advance file on an I/O device.	AVFIL4	
#7	WEOF	Write an End-of-File mark on an I/O device.	WEOF4	SPECIAL
#8	HOME	Normalize an I/O device.	HOME4	FUNCTIONS
#9	TERMINATE	Terminate operations queued to an I/O device.	TRMIN8	OR POSITIONAL
#A	ASSIGN	Assign logical file to an I/O device or another logical file.	ASSGN4	CONTROL
#B	TASSIGN	Test assignment of logical file and interrogate its assigned device	TASS4, DTASS4	
#C	IOWAIT	Wait for completion of one (or more) I/O operations.	UL13	
#23	TAKE	Take exclusive use of an I/O device.	TAKE4	
#24	GIVE	Give up exclusive use of an I/O device.	GIVE4	/

Table 1-1. REX I/O Services

All the REX I/O calls require the following information when the service is called.

- Name of the logical file
- Data format mode selected
- Error recovery mode selected
- Control mode selected (Wait/Quick Return Mode)
- Buffer mapping information address and size
- Device dependent options selected
- The service desired
- The service options desired

The User File Table (UFT) data structure contains the logical file name and options associated with an I/O operation. The argument list for a REX I/O call must contain:

- UFT address
- REX call number and options

The Assembly language programmer uses Register 2 for the UFT address and Register 8 for the Service Call number and options.

By providing a CAN-coded task name in Registers 12 and 13 and setting the OTF option in Register 8, a privileged task can reference another task's file assignments with the following REX services:

- READ
- WRITE
- REWIND
- BKFILE
- BKRECORD
- AVRECORD
- AVFILE
- WEOF
- HOME
- TERMINATE
- ASSIGN
- TAKE
- GIVE

The READ and WRITE I/O services can use Registers 14 and 15 to hold the buffer address and transfer count.

The FORTRAN programmer can perform device independent I/O without using REX calls directly. The FORTRAN Run-Time Library provides an execution time interface between a compiled FORTRAN program and the MAX IV REX services. The FORTRAN compiler generates linkages to the FORTRAN Run-Time Library that are satisfied at link edit or link load time.

The MAX IV Utility Library enables the FORTRAN programmer to utilize any function offered by the MAX IV REX services. The FORTRAN programmer can build a UFT and select the desired options using FORTRAN callable subroutines available in the MAX IV Utility Library. Additional I/O subroutines are available in the MAX IV Executive Functions and Process I/O Library (a standard MAX IV library) and the MODACS III Process I/O Subsystem (Model 8830).

Data is transmitted to and from memory data buffers within the addressing space of the calling task. The buffer address and size are arguments the user must pass with each READ or WRITE service call.

In a FORTRAN program, this data buffer may be transparent to the user but it is allocated and used by the FORTRAN Run-Time Library and resides in a labeled COMMON area generated by the FORTRAN compiler or in an area near the end of the task body.

In general, MAX IV allows the programmer to modify all or part of the UFT at any time after the I/O system has been called. The calling sequence code and the UFT must be considered to be "busy" until the I/O operation they initiate is successfully completed or terminated. Multi-level queuing requires multiple calls and UFT's.

A special option allows the programmer to request multiple I/O operations using the same User File Table (UFT) and/or buffer to one or more devices. This option requires that System Recovery is specified and event indications are ignored. These restrictions exist only because in this mode, it cannot be determined to which operation the error or event applies.

1.7.1 DATA TRANSFER SERVICES

The data transfer services (READ/WRITE) require, in addition, two arguments to specify the address of the data buffer and the buffer size (bytes). These arguments may exist as an extension of the UFT or in R14 and 15 (specified by a UFT option).

The QUICK option (R8 Bit Ø set) acts as a one-bit argument and specifies the Quick Return mode. Control returns to the caller as soon as the specified operation is started or queued. If not set (R8 Bit Ø =Ø), control is not returned to the caller until the operation is complete and error-checked (Wait mode). An example of a data transfer service is shown below:

EXAMPLE:

```
READ  EQU Ø                (REX Service #)
QUICK EQU #8ØØØ            (Data Value Specifying Bit Ø Set)
  .
  :
*START EXECUTION HERE
  LDI,R14 BUFER
  LDI,R15 8Ø
  LDI,R8  READ!QUICK
  LDI,R2  UFTA
  REX,MAXIV
*RETURN HERE AFTER OPERATION QUEUED
  .
  :
*BUFFER AND UFT IN NONEXECUTABLE AREA OF PROGRAM BODY.
BUFER RES 41,Ø              (Extra Word for End-of-Text Marker)
UFTA  DFC Ø                 (Status)
      DFC @BI               (File Name)
      DFC #BØØØ             (Options)
      DFC Ø                 (Device Position Index)
      DFC Ø                 (Byte Count)
      DFC Ø                 (File Table Pointer)
      DFC #4ØØØ             (Extended options (Buffer address
*                              and byte count in regs))
      DFC Ø                 (Mapping Info) (Ø=buffer in
*                              operand map)
```

1.7.2 POSITIONAL CONTROL SERVICES

The positional control services do not require any buffer arguments. The QUICK option (R8 Bit Ø set) selects Quick Return mode and causes the return as soon. as the operation is queued; otherwise, the return does not occur until the operation is complete.

1.7.2.1 HOME

A programmer planning to take advantage of the device-independent nature of the device handlers should HOME the device initially. This "normalizes" the device, clearing any initialization problems should another program have left the device in an unknown state or carriage position. This is ideal when the program uses the next available media position. Once the device has been normalized, data can be transmitted normally.

1.7.2.2 REWIND

The programmer should use REWIND instead of a HOME if the programmer wishes to start at a Beginning of Media (BOM) position.

1.7.2.3 WEOF

At the end of the program, the WEOF (Write End-of-File) function may serve as a convenient final action to close down the device used and mark the limits of the utilized medium.

1.7.2.4 Advance/Backspace

The other positional operations (ADVANCE/BACKSPACE) are for use in special applications and require device-dependent programming. Most devices implement the ADVANCE functions. Printing devices generally skip lines or cause form feeds (pages) with these operations, while secondary storage devices usually skip sequential physical records. Only a few devices implement BACKSPACE functions (magnetic tape and disc) since reverse media control is not always available.

BACKSPACE and ADVANCE operations to secondary storage devices transmit some data, but usually only enough to detect the presence of an EOF record. This data is buffered by the handler and does not require that the programmer supply a local data buffer, as do READ and WRITE operations.

1.7.2.5 TERMINATE

The TERMINATE service is similar in calling sequence to other positional control services, but this operation is not queued. It immediately dequeues (and terminates if necessary) all I/O operations that:

- Are in the queue for the file's assigned device
- Were queued by the calling task

If the ALL option is selected, (R8 Bit Ø set) all the operations queued for this device are dequeued even if not queued by the calling task. Only privileged tasks can perform the latter function.

1.7.2.6 Test Assign

The TEST ASSIGN service returns to the calling task immediately. Information relating to the file and its device assignments appears in registers or a snap block. If the file does not exist, the UFT error bit is set if the User Error Recovery option is specified in the UFT calling arguments.

1.7.2.7 ASSIGN

The ASSIGN service requires one argument in R15. The option bit specifies a default assignment if set, and the current assignment when not set. Default assignments are made for a file when it is desired that either a backup or a standard device be established. The second assignment can be moved to the current assignment by assigning the file to itself. A file can be closed (the FAT entry made vacant) by assigning it to a device name of Ø. This is allowed only for temporary files. A file is permanent if it was assigned at the time the task resources were defined. It is temporary if a vacant entry was assigned during task execution.

1.7.2.8 I/O WAIT

The I/O WAIT service allows a series of options to cause a task to suspend (I/O hold) after having initiated one or more Quick Return I/O requests. The task will be resumed after one or more specified I/O requests complete or when all operations currently in progress complete.

1.7.2.9 TAKE and GIVE

The TAKE and GIVE EXCLUSIVE USE services allow a task to exert exclusive control over a device. Requests for exclusive use may be requested as an immediate operation or as a queued request. This is useful if Quick Return mode is used in I/O operations, and a group of records must be written contiguously.

1.8 NAMING CONVENTIONS

1.8.1 FILE/DEVICE NAMING CONVENTIONS

I/O operations are performed on logical files and not directly on devices. The file acts as an intermediary between the user and the physical device, thus allowing I/O operations to be programmed so that the target physical device need not be selected when the program is written.

File-to-device or file-to-file assignments may be made through the REX ASSIGN service or may be made external to the coding of the program:

- By other programs
- By operator directive
- By Job Control directive
- By a cataloging
- By a system generation function

The user assigns each file a 1- to 3-character symbolic name that is unique to the task and includes this name in the User File Table (UFT) which accompanies each I/O call. The name is linked to the physical device through a File Assign Table not directly available to the user. Task files belong to a particular task and can only be addressed by that task. The files associated with the exceptional condition task may be addressed by other tasks and are said to be global files. Each file has a sequential File Position Index or number that is:

- Incremented for each forward operation
- Decremented for each reverse operation
- Cleared for each rewind or reassignment of the file to a device
- Copied on reassignment of the file to another file

This index is used as a direct address when the file is assigned to random access (disc) devices and as a simple counter when assigned to sequential access devices.

Files are assigned to "logical" devices or other files. Logical devices differ slightly from physical devices:

- If a physical device has its own unique controller or channel, it is also a logical device (for example, card reader, line printer). Some physical devices have several subdevices sharing a common controller/channel. These subdevices are referred to as transports.

- Randomly addressable devices such as disc packs may be logically partitioned into many logical devices. Any logical device may be either the file's current assignment or its default (or backup) assignment. Logical devices are always addressed by physical record (unit record sizes convenient to the device).

Logical device names are set at system generation. The following conventions are used for standard devices. The following naming conventions permit standardization of logical devices for system generation control information and documentation. All device names are expressed in CAN-code in the user's Assembly-language program.

1.8.2 DISC PARTITIONS WITHIN A PACK

A disc controller may have a single fixed or removable pack transport or a series of such transports. Most disc partitions are identified by the following 3-character name convention:

apx

"a" usually indicates the task name that exclusively uses the partition (B, C, D, ...) if such exclusive use is optioned. If any batch-processing task can use the partition, the letter "A" is used; and if any real-time task can use the partition, the letter "X" is used.

For user-defined partitions:

- "p" is the transport designator variable (usually a number Ø, 1, ...)
- "x" is a partition designator variable, that may be any character necessary to make the partition unique

EXAMPLES:

AØA
B1C
D2H
XØ5

Any three-character unique name can be used to name partitions. The names of the standard MAX IV Software System disc partitions do not use the pack-designator character described above but use names that more closely define their usage. These names are described in Appendix B of the MAX IV General Operating System, System Guide Manual. Generally, the last two characters of a standard partition are given the name of the logical file normally assigned to it. Thus, the disc partition normally assigned to the batch-processing Library file LB might have the name ALB or BLB, for example.

A physical device may contain more than one physical transport. For disc transports, the physical transports can be further subdivided into logical transports. Logical transports define a method of identifying separate media on a multi-media single-transport controller. Each logical transport can be looked at as if it were a separate pack.

A partition of a pack may be accessed by a user as if it were an independent device with a unique name. Partitions cannot overlap logical transport boundaries. A disc partition is an assignable logical device, while the disc controller or its transports are not directly addressable by the user. Each partition of a logical transport has its own offline/online error control.

1.8.3 STANDARD DEVICE AND CONTROLLER NAMES

Appendix B in the MAX IV General Operating System, System Guide Manual provides a list of the standard device and controller names used in MAX IV.

1.8.4 LOGICAL FILE NAME CONVENTIONS

Files are addressed by the programmer and have devices assigned to them. Certain logical files are standard in a MAX IV system. These files are available to all tasks (global) and usually have assignment restrictions. Such files are contained in the Global File Assign Table.

FILE NAME	NAME AND PURPOSE	ASSIGNMENT
OC	Operator-to-Computer File	TYn (or any input device)
CO	Computer-to-Operator File	TYn (or any printing devices)
DO	Diagnostic Output File	TYn, LPn (or any printing device)

The following global disc files are also standard:

LM	Load Module File with optional memory resident directory (any disc partition)
SM	Load Module File for non-resident OC directive overlays and other system modules (any disc partition)

Other permanent (global or resident task) files may be established at system generation time. Task files for use by a non-resident task are specified when the task is cataloged. Files may be created at run-time if vacant file table entries are specified at system generation time (global and resident task files) or cataloging time (non-resident tasks).

A complete list of all standard system global logical file names is included in Appendix C of the MAX IV General Operating System, System Guide Manual. Appendix D contains standard logical files.

1.9 USER FILE TABLE (UFT)

The UFT is an 8 or 10 word table assembled in the user's program. The table contains entry and return arguments associated with the REX service for standard device I/O operations. Process I/O devices require two additional arguments in the UFT.

Before an I/O REX service is performed, the user must have prepared five words in this table as entry arguments. These words are:

UFTNAM
UFTOPT
XUFOPT
XUFADR
XUFCNT

The UFT contains the name of the file, specifications that describe the format and options desired, and buffer mapping information.

After an I/O REX service is performed, the I/O system processes and returns information in the other words of this table (return arguments). Refer to Appendix A for an overview of the MAX IV I/O Data Structure Linkage. Appendix B shows the MAX IV UFT.

Table 1-2 shows a typical UFT as it would be coded in a user's Assembly language program. The FORTRAN Run-Time Library synthesizes a UFT in a labeled Common block so the user need not be concerned with this table. Other FORTRAN extensions allow the user to deal with a UFT directly in a FORTRAN program. Refer to the MAX IV Utility Library, LRM.

WORD	OPERATION	SET BY	PURPOSE
0 (0th)	DFC 0	BIOS	Status After Operation
1 (1st)	DFC @XYZ	User	File Name
2 .	DFC #A000	User	Formats and Options
3 .	DFC 0	User and BIOS	Device Position Index
4 .	DFC 0	BIOS	Byte Count
5 (5th)	DFC 0	BIOS	Pointer to Assign List
6	DFC 0	User	Extended options
7	DFC 0	User	Mapping information
8	DFC BUF	User	Buffer address (may be in R14)
9	DFC CNT	User	Byte count (may be in R15)

Table 1-2. Typical UFT (Assembly Code)

1.9.1 WORD 0 (STATUS)

Word 0 is set to zero by the I/O system after the REX service is called by the user. After the requested operation is performed, bits are set to indicate the error or event conditions that exist for the physical device to which the file is assigned.

The user's program may test bits in this word after the execution of the service and make decisions concerning the success of the operation or the occurrence of an event (EOF detected, et cetera). Recovery from certain errors is attempted automatically by the system if the user requests System Recovery in the option word below.

Word 0 has an "all inclusive" busy status bit (Bit 15) that, if set, means that the UFT is busy. It is set from the time an operation is queued until the operation has been successfully concluded. If a UFT has Bit 15 set when an I/O REX service is called, the I/O system assumes this UFT is busy from a previous I/O operation and relinquishes time until the operation is completed. This can be inhibited by a special option bit in Word 6 that causes the I/O system to still queue the operation.

Figure 1-6 shows the format of the UFT Status Word. If the bits are set (=1) the indicated conditions are true. Some device handlers use Bits 1 through 8 to extend the "events" that the handler can indicate. The user is cautioned to ensure that ER is set before assuming that Bits 1 through 8 have the indicated meanings.

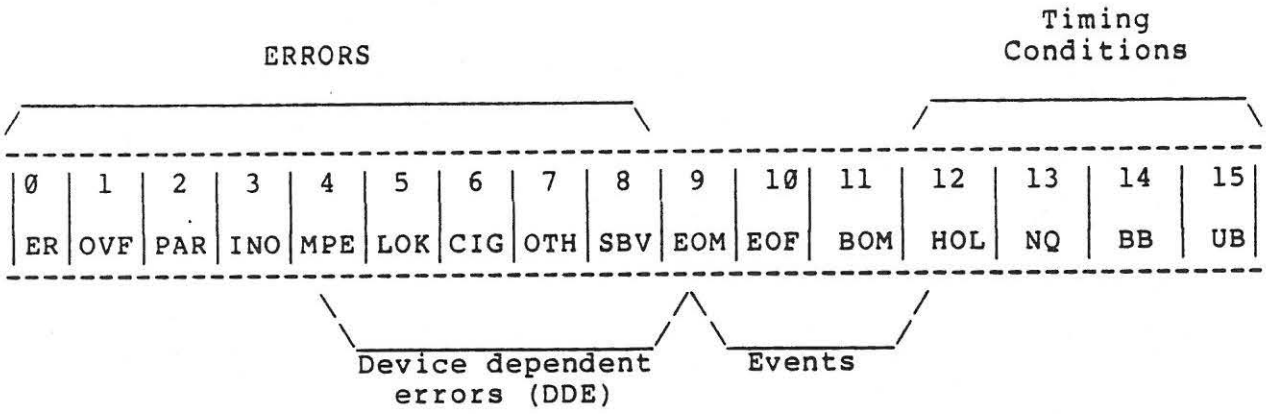


Figure 1-6. Status Word (#Ø) of UFT Returned by BIOS (UFTSTA)

When the NQ Bit is reset (=Ø), that is, when the requested I/O operation has been queued, the bits set (=1) in the status word have the following meanings:

Bit	Name	Description
Ø	ER	"Some error exists" (as per Bits 1-8 or NQ)
1	OVF	Underflow/Overflow or other data sync error
2	PAR	Data parity or checksum error
3	INO	Device inoperable or offline
7	OTH	If OTH is set and UFTBCT is Ø then the I/O operation was terminated.
9	EOM	End-of-Media (end of tape, paper low, tape low, last sector, device offline, etc).
1Ø	EOF	End-of-file/end of form $$.
11	BOM	Beginning of Media (at load point, top-of-form etc).
Timing Conditions are testable in Quick Return only.		
12	HOL	"Hold (by operator or first byte has not been transferred).

13	NQ	See explanation below.
14	BB	"Buffer busy" or last byte (word) not yet transferred.
15	UB	"UFT busy" or operation not yet completed.

The NQ Bit set indicates the requested operation has not been queued. NQ is set only when User Error Recovery is used (SR Option Bit Ø = Ø) or Return if Not Queued (RNQ Option Bit 7 = 1) is set. When the NQ Bit is set, Bits 1 to 11 of the status word take on special meaning as described below.

Bit	Name	Description
Ø	ER	Error!
1	OVF	File Search in progress/Logical Device busy.
2	PAR	No I/O node is currently available.
3	INO	Device inoperable or offline.
7	OTH	You have attempted an operation that would have aborted you under pure System Error Recovery procedures! The "WHO" and "WHY" error reason codes will be found in your UFTBCT and UFTFAT words, respectively.
1Ø	EXU	Exclusive use has been taken by another task.
12	HOL	"Hold" hardware status detected.
13	NQ	The NQ Bit must equal 1 for the bits above to have the indicated meaning. NQ indicates operation not queued.

1.9.2 WORD 1 (FILE NAME)

Word 1 is set by the user (as a call argument) as the three-character name of the logical file. This name must be expressed by a CAN (compressed-alphanumeric code) data constant, and it must be a valid name in the Task or Global File Assignment Table.

1.9.3 WORD 2 (OPTIONS)

Word 2 is set by the user as a call argument to describe the data mode and various options to be used when the I/O operation is performed. Bits are set in this word by the user to describe such conditions. A detailed schematic of this word is illustrated in Figure 1-7.

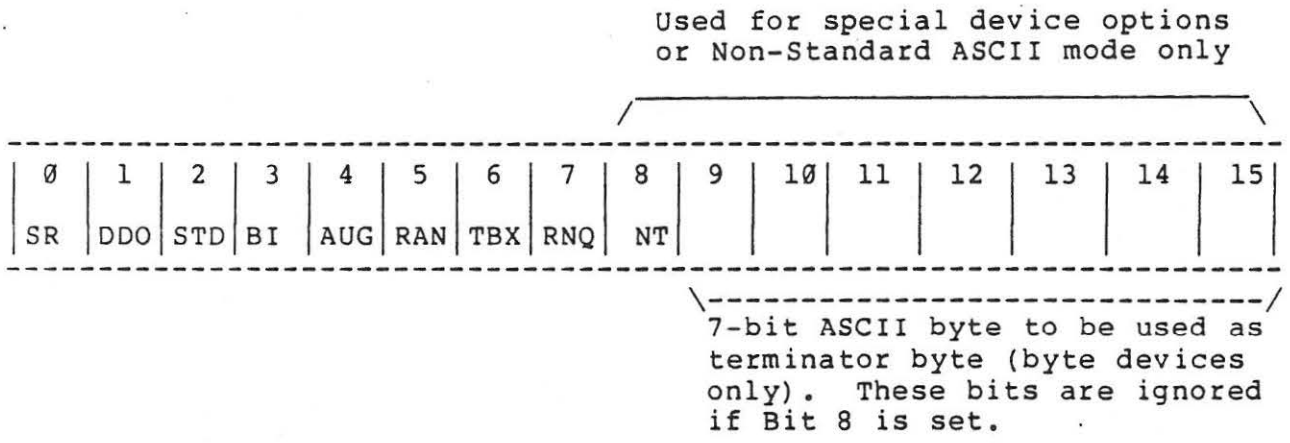


Figure 1-7. Option Word (#2) of UFT Specified by Programmer

The indicated operation/function will be performed/selected if the bit is set (=1) below:

Bit	Name	Description
0	SR	"System recovery" (automatic recovery if errors detected and will notify operator instead of returning to calling task).
1	DDO	Device dependent option (device functions unique to certain devices).
2	STD	"Standard format" (Non-Standard if 0) data format mode.
3	BI	"Binary mode" (ASCII mode if 0) data format mode.
4	AUG	Augmented device option (device dependent).
5	RAN	Random mode (random access devices only) use word #3 of UFT as 16 bit relative sector, or pointer to 32 bit random access address.
6	TBX	Inhibit data editing functions (trailing blank, suppression, et cetera).
7	RNQ	Return to user if I/O request Not Queued. If set in conjunction with Bit 0 reset, the I/O system will return control to the user when errors occur ONLY after failing to automatically recover the error... the device will NOT be placed Offline!

1.9.3.1 Error Recovery Mode Bits 0 and 7 - SR and RNQ

The Error Recovery mode is determined by the conditions of Bits 0 and 7 of Word 2 of the UFT. System Error Recovery is selected by setting Bit 0 equal to 1 in Word 2 of the UFT.

When Bit Ø equals Ø the user is responsible for error recovery. There are two types of User Error Recovery. Total User Error Recovery returns to the user task as soon as an error is detected. When Bit 7 of Word 2 of the UFT equals 1, an alternate User Error Recovery mode is selected. The I/O system will return control to the user when errors occur only after failing to automatically recover the error. Refer to Table 1-3.

Bit Ø SR	Bit 7 RNQ	DESCRIPTION
Ø	Ø	User Error Recovery, return immediately if errors occur.
Ø	1	User Error Recovery, return only after unsuccessful attempt to automatically recover any errors.
1	Ø	System Error Recovery, place device offline and notify the system operator after unsuccessful attempt to automatically recover any errors.
1	1	System Error Recovery as above except return control to me if the operation cannot be queued.

Table 1-3. Error Recovery Mode

1.9.3.2 Device Dependent Option - Bit 1 - DDO

Bit 1 (DDO) may be set in the UFT option word by the user only when the user knows that the logical file being used is assigned to a particular device. The presence of this bit causes some devices to perform functions not common to all devices but unique functions to that device. Usually, such functions are limited to special applications.

1.9.3.3 Data Format Mode - Bits 2 and 3 - STD,BI

Four basic modes of data transfer are specified by Bits 2 and 3 of the UFT option word (Word 2). These modes are summarized in Table 1-4 below and are discussed in detail in the chapter for a particular handler.

Bit 2 STD	Bit 3 BI	MODE
Ø	Ø	ASCII Mode } Non-standard modes (for
Ø	1	Binary Mode} device-dependent programming)
1	Ø	ASCII Mode } Standard MODCOMP modes (for
1	1	Binary Mode} device-independent programming)

Table 1-4. Data Format Modes

1.9.4 WORD 3 (DEVICE POSITION INDEX)

Word 3 specifies either a 16- or 32-bit position index. The normal "default" index is 16-bits and is contained in this word. If the 32-bit index is selected (option in word 6), this word contains the address of a double-word memory location which is to be used as the Device Position Index (DPI). This option is useful when a disc partition greater than 16K sectors is to be addressed in random mode. (In sequential mode, a 32-bit Index is automatically maintained by the system.) For privileged tasks, other addressing parameters are available to address disc devices and are discussed in the I/O section for the disc handler.

The Device Position Index is normally set to zero by the I/O system each time a REWIND function is performed, or when the file is "ASSIGNED" to a logical device. This word is incremented for each READ or WRITE or forward positional operation and decremented for each reverse positional operation. It keeps a count of the number of physical records encountered.

For special devices such as the disc, which may be randomly addressed, this word may also be set by the user as a call argument to the relative address (sector) of the random physical record to be used. When this sector address is not modified by the user, disc partitions act as "sequential" devices. Thus, a randomly addressable device may be sequentially or randomly addressed.

The random function is ignored for sequential devices. Whenever the random bit is set, the Device Position Index, used by the device handler, is incremented upon successful completion of the operation and is returned to the UFT and to a similar cell in the file's entry in the File Assignment Table.

1.9.5 WORD 4 (BYTE COUNT)

After a READ or WRITE operation is performed, Word 4 is set by the system to the number of bytes actually transferred by the assigned device (non-data chaining only) regardless of the number of bytes specified by the user. This byte count does not exceed the size of the device's physical record (if fixed record size) or the user's specified data buffer, whichever is smaller.

If the number of bytes in a physical record is greater than the specified buffer size during Standard Binary READ operation of byte-type devices (for example, PR and CR), the overflow/underflow (OVF) error bit and the hold (SBV) bit are set in the UFT status word, but the error (ER) bit is not set. Most standard DMP devices have an even number of bytes per record transmitted except when certain device-dependent characters are detected in the data stream.

1.9.6 WORD 5 (ASSIGN LIST POINTER)

Word 5 is set by the I/O system for its own use and aids in a low-overhead response to each file's I/O operation. The contents of the word are determined after the first file operation using each UFT is performed, and contains an index that links the file name to its entry in the Task File Assign Table or the Global File Assign Table.

Each subsequent I/O operation using this UFT can be performed without a sequential search of the appropriate File Assign Table. The user should not be concerned with the contents of this word. This pointer is set by any file operation (for example, HOME, REWIND) that uses a UFT for calling arguments.

1.9.7 WORD 6 (EXTENDED OPTIONS)

Word 6 is used to specify other characteristics about an I/O operation not available in Word 2. A detailed schematic of this word is illustrated in Figure 1-8. The right byte of this word is device dependent and is illustrated in the chapter for each handler.

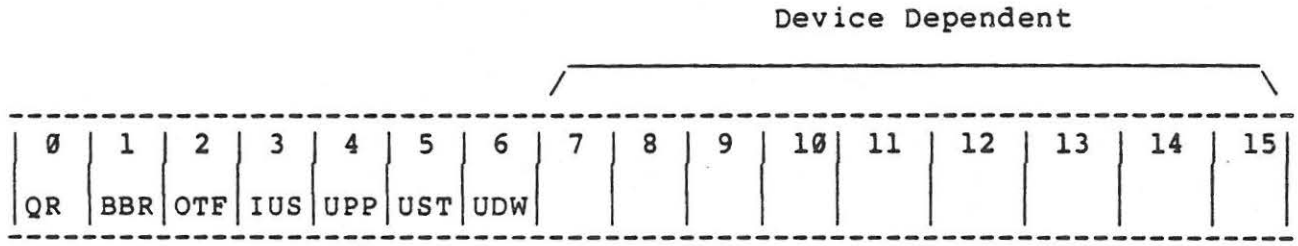


Figure 1-8. Extended Option Word (#6) of UFT Specified by Programmer

Bit	Name	Description
0	QR	Quick Return operation.
1	BBR	Buffer address and Byte Count are in R14/R15.
2	OTF	Use another task's File Assign Table (FAT) to resolve the file-to-device assignment. Privileged operation only. Other task's name in R12/13 (1-to-6 character CAN-code).
3	IUS	Ignore UFT busy status when I/O system is entered (queue anyway).
4	UPP	Use 32 bit Position Index (Word 3 points to double-word memory location where index is maintained).
5	UST	User specified timer is in Register 3.
6	UDW	User defined words prefix the UFT.

1.9.7.1 Quick Return - Bit Ø - QR

Quick Return mode can be selected by setting Bit Ø equal to 1 in Word 6 of the UFT. Quick Return can also be specified as an option in Register 8 on the REX call. This bit set in either place results in Quick Return mode. Both bits do not have to be set.

1.9.7.2 Buffer Address and Byte Count - Bit 1 - BBR

When Bit 1 of Word 6 of the UFT is set equal to 1, the I/O system uses the value in Register 14 as the buffer address and the value in Register 15 as the byte count (instead of UFT Words 8 and 8).

1.9.7.3 Other Task's File - Bit 2 - OTF

When Bit 2 of Word 6 of the UFT is equal to 1, it indicates to the I/O system that the logical file name in Word 1 should be associated with the File Assign Table of the Task name contained in Registers 12 and 13. This feature is available only to privileged tasks.

1.9.7.4 Ignore UFT Busy - Bit 3 - IUS

This option is selected by setting Bit 3 of Word 6 of the UFT equal to 1. This feature allows multiple I/O operation requests to be processed using the same UFT and/or buffer to one or more devices. This option requires System Recovery.

1.9.8 WORD 7 (MAP IMAGE)

Word 7 is used to provide the I/O system mapping information about the data transfer. Normally, the operand map is the map used for data transfers and is selected by this word being zero. If the instruction map is to be used to resolve the virtual-to-real addressing during the transfer, this word is set to -1 (#FFFF). If the calling task is a one map task, these two parameters will yield identical results.

For privileged tasks, this word may specify an actual page number that contains a map image to be used for the transfer. This is useful if the programmer wishes to transfer data into another task's memory or to initialize some memory not mapped into the calling task's addressing space.

1.9.9 WORD 8 (BUFFER ADDRESS)

Word 8 may be specified in Register 14 if desired, in which case, it may be eliminated from the UFT. This word contains the address of the programmer's I/O buffer in the map selected by Word 7. If data chaining is desired, this word contains the address of the chain list in the map selected by Word 9.

1.9.10 WORD 9 (BYTE COUNT)

Word 9 may be specified in Register 15 if desired, in which case, it may be eliminated from the UFT. This word contains the byte count. For non-chaining operations, this is always the maximum number of bytes desired, specified as a magnitude no greater than #8000.

If data chaining is desired, this word contains mapping information for the chain list address given in Word 8. The chain list may be either in MAP 0 or in the map used for the data transfer (specified by Word 7). To indicate this, Word 9 contains -1 (#FFFF) or -2 (#FFFE) respectively. Any value between #8000 and #FFFE exclusive in this word constitutes a parameter error.

1.10 TESTING STATUS AFTER COMPLETION OF I/O OPERATION

After completion of an I/O operation, Word #0 (STATUS) in the UFT contains bits that may be tested. In the Quick Return mode, some of these bits may be tested during the I/O operation. This section discusses those conditions under which these bits must be tested and to what level of detail for particular requirements. The user should refer to Figures 1-6 and 1-7 during the following discussion.

1.10.1 SYSTEM ERROR RECOVERY

If the System Recovery Option is set (Bit 0, Word 2, UFT) and if Wait mode is used, all unrecoverable error conditions are reported directly to the system operator after a device-dependent number of retry operations has been attempted. Refer to the I/O ERROR MESSAGES, Section 1.11. The operator may attempt to correct the problem and cause the peripheral to be returned to operation without the calling task being aware of the problem. Likewise, in the Wait mode, the timing conditions (Bits 12, 13, 14, and 15) occur while the task is suspended, thus never needing to be tested by the task.

Bits 0 (ER), 13 (NQ), and one of the bits shown in Figure 1-6 are set in Word 0 of the UFT (UFTSTA) and the I/O system immediately returns to the caller under the following set of conditions:

- If the RNQ (Return if Not Queued) bit is set and/or User Error Recovery is specified.
- And if the operation cannot be queued for one of the following reasons.
  - The device is offline.
  - The device is inoperable.
  - The device is under exclusive use.
  - The hopper is empty or the stacker is full on the card reader.
  - A hold exists on the card reader or line printer.
  - A low tape condition exists on the paper tape punch.

Only the event conditions (Bits 9, 10, and 11) need be interrogated by such a user after the completed I/O operation causes control to be returned to the caller.

When the System Recovery Option is set (Bit 0, Word 2, UFT) and the Quick Return mode is used, the operation is similar to above in that error conditions, unresolved after a device-dependent number of retry operations, are reported to the system operator. However, in the Quick Return mode, the calling task may test Bit 12 (Hold), Bit 14, (Buffer Busy), or Bit 15 (UFT Busy) at any time after the I/O operation has been queued but not completed. During this period, event or error conditions may not be valid. Only after the UFT BUSY bit is reset can these error or event conditions be tested with confidence. The NOT QUEUED (Bit 13) is not set if the System Recovery Option is set unless the RNQ (Return if Not Queued) bit is also set (see Wait mode, above).

1.10.2 USER ERROR RECOVERY

When it is necessary for a task to recover from I/O error conditions without delay, (for example, ignore errors or substitute another device), the user may make I/O calls in the Quick Return mode (or Wait mode) and with the System Recovery Option reset. When this is done, the first test that must be made by the programmer after the I/O system returns control is Bit 13 (NOT QUEUED). If this bit is set, the operation could not be queued because the device was offline or inoperable or the user's calling sequence was unresolvable (program error).

This condition also occurs if temporary exclusive-use has been obtained for the desired device on behalf of some other task. If the NOT QUEUED condition is found reset, the operation may still require testing for error. For a completed READ or WRITE operation, the data has been transferred but may be in error. The user may choose to perform his/her own retry operations or ignore such errors (if necessary because of timing restrictions). Neither errors nor events can be tested with confidence until the UFT BUSY bit is reset. This condition is always reset before return to the user if Wait mode is used. When timing conditions are tight, the user may start processing data in a buffer when the "buffer busy" bit is reset and before the UFT busy bit is reset, if the user is willing to test for errors later, or "after-the-fact".

The user can take advantage of the System Error Recovery feature of automatically retrying the operation when errors occur and the User Error Recovery feature of notifying the calling program of unrecoverable errors instead of the system operator and simply placing the device offline. This is accomplished by resetting SR and setting RNQ. These two bits specify a 2-bit Error Recovery Mode which is described in Section 1.9.3.1.

1.11 I/O ERROR MESSAGES

1.11.1 ONLINE MESSAGES

A programmer on the MAX IV system can choose to have all errors reported to the user program or may specify that all errors be reported to the system operator after a device dependent number of retry operations. The System Error Recovery frees the programmer of extensive error checking procedures, placing the burden on the system operator instead.

Three types of errors are reported to the system operator in the form of messages to the CO file (usually assigned to the console TY device):

1. Hardware errors or holdup conditions detected when the operation is requested by the user, but before interrupts are enabled and the operation is queued.

2. Hardware or program errors detected after I/O operation has been started or queued (for example, at the data or service interrupt levels when the device is terminated and tested for error).

3. Program errors that are unrecoverable.

The first type of error or condition causes the I/O system to force the task to relinquish time to lower tasks until the condition is corrected. If specified at system generation, (refer to the MAX IV SYSGEN, System Guide Manual) an optional message is output when the condition is detected. This message is called an I/O Stall Message and is in the following format.

MESSAGE	MEANING
XXX?/TTTTTT/000000	Device is in HOLD condition, is inoperable, does not respond, or has been put OFFLINE by the system or the operator.

where XXX is the logical device name followed by the system message NVM.(no volume mounted), or TTTTTT as the calling task's name, and 000000 as the current task overlay name. If no task or overlay name is output, the Operator Communication task queued the operation.

When the RNQ (Return if Not Queued) option bit is set, the I/O system immediately returns to the user without typing this error message (thus not stalling the user); Bit 0 (ER) and Bit 13 (NQ) of UFTSTA Word 0 are set regardless of Wait/No-Wait or System-Recovery/No-System-Recovery modes.

1.11.2 OFFLINE ERROR

The second type of error is detected at interrupt levels and may occur asynchronously with respect to the user's program that requested the operation. In this case, the logical device is placed in the OFFLINE state and requires an operator communications directive to subsequently cause the operation to be attempted again or ignored.

During the intervening period, all processing is not necessarily suspended for the task which requested the operation, but the task will eventually become wait-bound when it tests for the completion of the I/O operation that caused the device to go OFFLINE. If the task exits before testing, the EXIT service waits for the I/O operation to be successfully corrected by the operator.

This type of error causes the following class of messages, where XXX is the device name, and YYYY is the device status (as modified by the I/O system). Table 1-5 lists all OFFLINE error messages.

MESSAGE	REASON
!XXX OFFLINE OVF #YYYY	Data Sync Error Overflow/Underflow.
!XXX OFFLINE PAR #YYYY	Device Data Check Error or Parity Error.
!XXX OFFLINE INO #YYYY	Device Inoperable.
!XXX OFFLINE MPE #YYYY	Memory Parity During DMP I/O Operation.
!XXX OFFLINE LOK #YYYY	Attempted WRITE in Protected Area.
!XXX OFFLINE CIG #YYYY	Character in Gap - Unrecoverable Media Failure (for example, loss of disc sector formatting (prep)).
!XXX-OFFLINE OTH #YYYY	Device-Dependent Error.
!XXX OFFLINE HOL #YYYY	DMP (Direct Memory Processor) failure to transmit required block size or unrecoverable program error detected by Service Interrupt handler.
!XXX OFFLINE OFF #YYYY	Operation that caused failure has been removed ABORT service.

Table 1-5. OFFLINE Error Messages

A report of all I/O device operations currently in the logical device slough queue of device XXX is typed. This report is illustrated in the MAX IV Operator Communications, System Guide Manual.

Recovery from any of the above errors is device dependent. In this case, the physical device may have lighted error indicators that help the operator to uniquely define the problem. For example, if the card reader is examined after an error is detected, it should be clear from its indicators whether the last card read should be repositioned in the hopper for re-reading (motion error) or left in the stacker (non-pick). When the device has been made ready, the device can be returned to operating state by the following operator communications response:

```
/ON XXX
```

The above entry places device XXX online and causes the previous operation(s) to be repeated and task TTTTTT to be resumed. If the device continues to produce errors, the only alternative may be for the operator to abort the task or force continuation as if the operation had not been in error. The following directive causes device XXX to be put online and the I/O system is fooled into believing that the operation was completed successfully without error. Of course, the task is at the mercy of the possibly erroneous data.

```
/ON XXX NO
```

The parameter "NO" indicates that NO RETRY will be attempted.

1.11.3 ABORT MESSAGES

The third type of error usually results in the calling task being aborted.

```
!ABORT (YYY ZZZ #hhhh) /TTTTTT/OOOOOO
```

-where YYY and ZZZ are reason codes, #hhhh is the hexadecimal address of the I/O call causing the abort and /TTTTTT/OOOOOO is the task/overlay name. Abort reason codes for the I/O system are listed in Appendix E of the MAX IV General Operating System, System Guide Manual. A group/unit may be used when it is necessary to identify a particular controller.

CHAPTER 2
WRITING HANDLERS

Each MAX IV I/O handler consists of the directly connected DATA and SERVICE interrupt routines and a TASK level routine. The Task Level portion of the handler provides the housekeeping requirements that are necessary for device processing and the functions necessary to ensure device independence. Any functions normally considered as high overhead processes are best implemented here since these functions occur at the priority of the calling task.

The service interrupt routine initiates data transfer operations and processes terminations of all operations. Once many operations are queued for a particular controller, this interrupt routine processes all the operations without any assistance from the original calling task.

The data interrupt routine transmits the actual data (word or byte at a time) required during an operation between service interrupts.

BIOS provides several common subroutines to perform handler interface functions. Sections 2.2, 2.3, and 2.4 provide descriptions of these routines.

The naming convention used with these common subroutines readily identifies their processing level. Task level routines are named IT$xxx, SI level routines IS$xxx, and DI level routines ID$xxx.

2.1 GETTING FROM A REX TO THE HANDLER

Execution by the central processor of any REX instruction (op-code #23) results in the activation of the priority level 4 interrupt (unimplemented instruction trap). This is accomplished by saving the current PR (program register) and PS (program status) in locations #28 and #49, and loading the PR and PS with the address and status of the interrupt routine, which are permanently stored in locations #29 and #48. The interrupt routine then stores a slightly modified copy of the calling tasks PR and PS onto the task's PS/PR stack for return from a REX service.

This provides a mechanism to reduce the priority level. Normally, when an interrupt is cleared with a CIR or CAR instruction, the PR and PS are loaded with the contents of the dedicated locations associated with the highest active interrupt (for level 4, we have seen these are locations #28 and #49). Thus, if we construct a new PR and PS to pick up the address and status of the appropriate REX routine and then clear the interrupt, the CPU will be in the task state, executing a system routine, and a mechanism to return to the user will be available.

The new PS is constructed using the calling task's general register block, but with the privileged bit set and MAP Ø selected as both IM (instruction map) and OM (operand map). The new PR is determined by examining the actual REX instruction in the calling tasks IM. The REX number determines the routine to be entered. All REX calls of interest to a device handler enter BIOS (Basic Input/Output System). Finally, when I/O processing is completed and control should be returned to the user, the return PS and PR are removed from the task's PS/PR stack, level 4 interrupt active is set, and a CIR instruction is expected in order to return to the calling program.

In BIOS, the user's registers are pushed into a stack in the user's TCB (Task Control Block). Also pushed into this stack are a copy of the UFT, in extended form, and the REX request with options. If the logical file list entry in the UFT (UFTFAT) is Ø, then the file assign list whose address is specified in the TCB is searched for a name to match the name given in the UFT (UFTNAM). If no match is found, then the global file index, whose address is given in the Upper Control Block (UCB), is searched. If no match is found, the task is aborted for an illegal file name. If a match is found, the address of the matching entry is saved in the UFT. Next, it is determined what device the logical file is assigned to by following the file-to-file links (if necessary) until a device is found. If the file is not assigned to a device, the task is aborted for an unassigned file.

If the REX request was a TASSIGN (#B), a branch to the appropriate routine is made. Next, if the file is assigned to the "NO" device, the REX request is honored by returning EOF and putting $$ in the user's buffer for all READS, and updating record count in the UFT for all requests. If the file was not assigned to the "NO" device, and if the REX request is a TERMINATE(#9), GIVE+IMMEDIATE(#2Ø24) or TAKE+IMMEDIATE(#2Ø23), a branch is made to the appropriate routine. Otherwise, an IONODE is allocated for the calling task. Finally the relevant information is copied from the UFT and TCB into the IONODE, and a branch is made to the task level portion of the device handler.

2.2 TASK LEVEL SYSTEM ROUTINES

The following routines are to be added at a later date:

IO$ABO
IO$PRB
IO$WAM

IT$PHY

IT$PHY compares the transfer count specified in the IONODE with the maximum block size specified in the status word of the LDT. The routine returns the smaller size in registers 3 and 4.

Calling Sequence:

```
R1 = Address of IONODE

BLM,R6          IT$PHY
```

Results:

```
Registers used = R3, R4, R5
R3 = Transfer count in words
R4 = Transfer count in bytes (may be odd)
R5 = Device maximum size in words
```

IT$INI

IT$INI sets up the transfer count for non-DMP devices. It first calls IT$PHY to check the count given in the IONODE with the device limit. If the UFT specifies Standard Binary as the file type, and the REX request is a WRITE, this routine checks the buffer to ensure the first byte is a #Ø3 or #Ø7. If the first byte is not a #Ø3 or #Ø7, the calling task is aborted. The second word of the buffer is also checked to ensure the transfer count given in the buffer is less than or equal to the transfer count calculated by IS$PHY. If the count is greater, then the task is aborted. The resulting transfer count is placed in the IONODE.

Calling Sequence:

```
R1 = Address of IONODE

BLM,R9          IT$INI
```

Results:

```
Registers used = R3, R4, R5, R6, R7
R3 = Address of LDT
Routines called = IT$PHY
NODCNT (transfer count kept in IONODE) = true transfer count
```

Alternate Exits:

```
IO$ABZ - (R13 = @ICB) If Standard Binary WRITE of illegal
         buffer.
IO$ENO - If byte count (after reductions) is zero.
```

IT$DNI

IT$DNI sets up the transfer count for Direct Memory Processor (DMP) devices. It first determines, from the value stored in NODCNT, if a string chained transfer is requested. If string chained, it determines whether the O/S map image or the task's map image is selected. If chaining map is selected, the task is aborted unless it is privileged. If a single transfer is requested, IT$PHY is called to check the requested length with the maximum allowed by the device. Finally, the negative word count or the chain and map bits are stored in NODCNT.

Calling Sequence:

R1 = Address of IONODE
R13 = Address of program status word (PS)

```
BLM,R9          IT$DNI
```

Results:

Registers used = R3, R4, R5, R6
R3 = Address of LDT
Routines called = IT$PHY (only if single DMP transfer)
NODCNT (transfer count kept in IONODE) = DMP transfer count or chain map select.

Alternate Exits:

IO$APV - (R14 = @CHN) If non-privileged task attempts chaining.
IO$CNO - If NODCNT is zero at entry.

IT$CRC

IT$CRC interprets carriage control for Standard ASCII files. It picks up the first byte of the user's buffer, and if the byte is not hardware carriage control (that is, bit Ø is not set), then the byte is matched against each of the standard carriage control characters [(+),(-),(1),(Ø),( )]. If a match is found, then the device dependent value specified by the inline arguments is returned in Register 8. If no match is found, the device dependent value for space is returned. If hardware carriage control was specified, Register 8 is set to Ø and the specific byte (with the most significant bit reset) is returned in Register 9.

Calling Sequence:

R1 = Address of IONODE

```
BLM,R7          IT$CRC
DFC             *VALUE RETURNED IF "+" *
DFC             *VALUE RETURNED IF "-" OR "1" *
DFC             *VALUE RETURNED IF "Ø" *
DFC             *VALUE RETURNED IF " " *
```

Results:

Registers used = R7, R8, R9, R11
R8 = device dependent value or Ø
R9 = if carriage control was specified by user (sign bit of initial byte was 1), then R8 = Ø and R9 = user specified carriage control (initial byte without the sign bit).

IT$GPI

IT$GPI retrieves the File Position Index. If the RANDOM bit (Bit 5) of the UFT option word is set or a special disc mode is selected (XUFOPT bits 14 and 15), the File Position Index is retrieved from the UFT; otherwise, it is retrieved from the File Assign Table entry.

Calling Sequence:

```
R1 = Address of IONODE
R2 = Address of UFT stored in stack map in Ø

BLM,R9          IT$GPI
```

Results:

```
Registers used = R7, R14, R15
R14 = most significant half of position index
R15 = least significant half of position index.
```

IT$PPI

IT$PPI will replace a File Position Index in the UFT (both the calling task's UFT and the system copy in MAP Ø) and the FAT (File Assign Table entry) with the File Position Index modified to proper format (if necessary) given in R14-R15. If, however, XUFOPT Bit 14 (#E) indicates "absolute mode", no action is taken. IF UFTDPI (in user's map) is Ø, then UFTDPI is not used to address a 32-bit DPI.

Calling Sequence:

```
R1  = Address of IONODE
R2  = Address of UFT stored in stack in MAP Ø.
RIØ = Address of UFT stored in calling task's map.
R13 = Address of calling tasks PS.
R14, R15 = 32 bit File Position Index.

BLM,R9          IT$PPI
```

Results:

```
Registers used = R11, R7
File Position Index given in R14-R15 is put in UFT and FAT (in
     proper format).
```

IT$IPI, IT$DPI, IT$ZPI

IT$IPI increments by one the File Position Index by calling IT$GPI to get the index, adding one to the index, and calling IT$PPI to put back the result. IT$DPI is similar to IT$IPI except that it decrements the index by one and returns without storing the result if the index was zero or negative. IT$ZPI calls IT$PPI with a zero positon index.

Calling Sequence:

```
R1  = Address of IONODE
R2  = UFT address in stack in MAP Ø.
R1Ø = Calling task's UFT address in own map.

BLM,R9          IT$xxx    Where xxx = IPI, DPI, or ZPI.  B
```

Results:

```
Registers used = R7, R9, R11
Condition code = Z if at BOM after update.
               = N if IT$DPI called when index already zero or
                 negative - result not stored.
R14/15         = 32-bit resultant index.  If negative, IT$DPI
                 did not store it.
```

IT$AVR, IT$AVF

These routines set up the IONODE with the parameters necessary for a file positioning (advance file or advance record). The parameters are:

```
NODLEN = Ø (MAP Ø)
NODMAP = 1 (MAP Ø)
NODCNT = 2
NODBUF = PDT + 15 (PDTEOF)
STD (Standard) is set in NODOPT.
If AVF, then the FS (file search) bit is set in NODSTA.
```

Calling Sequence:

```
R1 = Address of IONODE
R3 = Address of LDT

BLM,R9          IT$AVF (or IT$AVR)
```

Results:

```
Registers used = R7
IONODE modified as shown above.
```

IT$WEF, IT$WEO

These routines set up the IONODE with the parameters necessary to output an internal buffer (SPACE, STX, $, $ or $,$) as a file mark. The parameters are:

NODLEN = Ø (MAP Ø)
NODMAP = 1 (MAP Ø)
NODCNT = 4 or 2
NODBUF = Address of the appropriate internal buffer
STD (Standard Mode) is set in NODOPT.
EOF (End-of-File) is set in NODEST.

Calling Sequence:

R1 = Address of IONODE

BLM,R9 IT$WEF (or IT$WEO)

Results:

Registers used = R7.
IONODE modified as shown above.

IT$CKL

IT$CKL determines if the data buffer will fit in the mapping range of the calling task. The address and length of the buffer are taken from NODBUF and NODCNT which are taken from arguments of the REX call.

If the buffer will not fit and System Recovery was specified in the UFT, the task is aborted; otherwise, a return to the calling task occurs with the MPE bit (memory parity error - which is used as a catch all error indication by the I/O system) set in the status word of the UFT.

Calling Sequence:

R1 = Address of IONODE

BLM,R9 IT$CKL

Results:

Registers used = R7, R8

IT$DAD

IT$DAD converts the position index supplied by the UFT or FAT to the appropriate sector, head and cylinder required by all disc handlers. The results are returned in Registers 14 and 15. Privileged users are allowed to specify "absolute" or "hardware" modes: Absolute mode specifies a position index relative to the entire physical or logical transport. Hardware mode specifies the sector, head, and cylinder to be used without modification.

Calling Sequence:

```
R1  = Address of IONODE
R2  = Address of pushed UFT
R3  = Address of LDT
R13 = Address of saved PS word

BLM,R9          IT$DAD
```

Results:

```
Registers used = R4, R5, R6, R7, R8, R14, R15
R14 = Left byte - sector number, Right byte - head number.
R15 = Cylinder number.
```

Alternate Exits:

```
IO$ABT (R13=@FPI) Illegal FPI for partition.
       (R13=@STC) Illegal sector, track, or cylinder.
```

2.3 SERVICE INTERRUPT LEVEL SYSTEM ROUTINES

The following routines are to be added at a later date:

IO$ABO	IS$SCD	IS$EXU
IO$PRB	IS$EOB	IS$FLS
IO$WAM	IS$DPI	IS$REQ
IS$XIN	IS$CSD	IS$DMP
IS$XAB	IS$CSH	IS$FTA
IS$RTN	IS$SNU	IS$TAK
IS$TEB	IS$RST	IS$GIV
IS$TER	IS$ERR	IS$PXU
IS$TPT	IS$OFL	IS$TXU
IS$POL	IS$NTY	

IS$WHY

IS$WHY determines the causes of a service interrupt by examining the PDT status word, the IONODE status word and the error status word in the IONODE (NODEST). After determining the cause, a branch is made to the appropriate routine specified by inline arguments. Routines already exist to handle most "SI" conditions. Generally, the first thing that the "SI" level of a device handler does is call this routine. If this routine is called first, there is no need to initialize R1 since there is a very short "DRIVER" routine that is executed first after a service interrupt that initializes R1. On a call to this routine, R1 must contain the address of the PDT minus 2 for the device that generated the service interrupt.

Calling Sequence:

```
BLM,R2          IS$WHY
DFC             *ADDRESS OF PDT QUEUE EMPTY ROUTINE*
DFC             *ADDRESS OF ERROR FREE SHUTDOWN ROUTINE*
DFC             *ADDRESS OF PASS2 STARTUP ROUTINE*
DFC             *ADDRESS OF ERRORS DETECTED SHUTDOWN ROUTINE*
DFC             *ADDRESS OF DEVICE OFFLINE ROUTINE*
DFC             *ADDRESS OF DEVICE OFFLINE - NO RETRY ROUTINE*
DFC             *ADDRESS OF DEVICE UNDER EXCLUSIVE USE ROUTINE*
DFC             *ADDRESS OF SHUTDOWN BUT FILE SEARCH ROUTINE*
```

*NORMAL RETURN INDICATES STARTUP CONDITION

A Typical Call To IS$WHY Looks Like

```
BLM,R2          IS$WHY
DFC             IS$XIT
DFC             IS$CSD
DFC             Ø * EXCEPT TO TTY DEVICES
DFC             IS$ERR
DFC             IS$OFL
DFC             IS$NTY
DFC             IS$EXU
DFC             IS$FLS
```

Results:

Registers used = R1, R2, R3, R4, R5, R6, R7, R8, R9

If normal return (if startup "SI"), then;
- R1 = Address of PDT
- R3 = Address of IONODE
- Routines possibly called: IS$XSI, IS$TXU

IS$SIO, IS$RTY, IS$RKB

IS$SIO is a routine that will initialize the PDT with all the information necessary to perform an I/O operation. UFT status bits are also set to indicate startup conditions. The routine exits through inline arguments, based on the type of I/O call. This routine is generally called immediately after IS$WHY. In the PDT, the following words are set:

PDTSTA — Only controller busy and shutdown bits set. The shutdown bit is set so that the next SI will be shutdown SI.

PDTTIM — Set to Ø.

PDTUAP — A routine (IS$UMA) is called to determine whether the calling task's UFT is in MAP Ø or in the calling task's operand map. In either case PDTUAP is set to the proper map image actual page.

The following words are initialized only if device is not a DMP device:

PDTCAD — Set to buffer address that was saved in the IONODE.
PDTTBC — Set to Ø.
PDTBTC — Set to Ø.
PDTGAP — Set to Ø.
PDTBOC — Set to Ø.
PDTLFC — Set to Ø.
PDTBCN — Set to transfer count stored in the IONODE.

IS$RTY is identical to IS$SIO except that it does not reinitialize the watchdog timer (PDTTIM).

IS$RKB will only reset the UFT status word to indicate startup and exit through the inline argument. This is useful for pass 2 type operations.

Calling Sequence:

All three routines have identical calling sequences.

R1 = Address of PDT
R3 = Address of IONODE

```
BLM,R2          IS$SIO (OR IS$RTY OR IS$RKB)
DFC             *ADDRESS OF READ ROUTINE*
DFC             *ADDRESS OF WRITE ROUTINE*
DFC             *ADDRESS OF REWIND ROUTINE*
DFC             *ADDRESS OF BACKSPACE FILE ROUTINE*
DFC             *ADDRESS OF BACKSPACE RECORD ROUTINE*
DFC             *ADDRESS OF ADVANCE RECORD ROUTINE*
DFC             *ADDRESS OF ADVANCE FILE ROUTINE*
DFC             *ADDRESS OF WRITE END OF FILE ROUTINE*
DFC             *ADDRESS OF HOME ROUTINE*
DFC             *ADDRESS OF TAKE EXCLUSIVE USE ROUTINE*
DFC             *ADDRESS OF GIVE EXCLUSIVE USE ROUTINE*
```

Results:

Registers used = R4, R5, R6, R7, R8
Routines called = IS$UMA
Unless called as IS$RKB, the PDT is initialized as described above and the registers have the following value:

R1 = Address of PDT
R3 = Address of IONODE
R4 = Address of UFT position index.
R5 = Calling task operand map actual page number (MIAP).
R6 = Address of LDT.
R8 = Device configuration word.

IS$UMA

IS$UMA will determine if the UFT of the calling task is in MAP Ø (due to a call from a system service) or in the task's operand map. Then it will set the proper map image actual page in the PDT.

Calling Sequence:

R1 = Address of PDT
R3 = Address of IONODE

```
BLM,R2          IS$UMA
```

Results:

Registers used = R5
PDTUAP set to proper Map Image Actual Page
R5 = Set to proper Map Image Actual Page

IS$PDI

IS$PDI initializes the gap control words and line feed counts in the PDT. If should be called by any device handler requiring line feed counts or gap control. The following words in the PDT are initialized:

PDTGAP The negative of the trailing count.

PDTBOC The negative of the leading count.

PDTLFC The negative of the carriage control byte in NODCCC unless the special bit (bit 4) of NODCCC is set, in which case it is the non-negated carriage control byte.

PDTSTA The carriage control bits (bits Ø-7), are logically "OR"ed with bits 8-15 of PDTSTA.

Calling Sequence:

```
R1 = Address of PDT
R3 = Address of IONODE

BLM,R2          IS$PDI
```

Results:

Registers used = R7, R8
PDT gap control and carriage control words initialized.

IS$ERO

IS$ERO merges the hardware status (given in R5) into the IONODE error status word NODEST.

Calling Sequence:

```
R1 = Address of PDT
R3 = Address of IONODE
R5 = Hardware status (the result of an ISx,R5,U instruction).

BLM,R2          IS$ERO
```

Results:

Registers used = R5
Device status (with the exception that the error bit is toggled) is "OR"ed into NODEST.

IS$SOL, IS$$EX

IS$SOL is a routine that processes device offline conditions if the device was offline before the I/O operation began. If the I/O call was made with System Recovery specified in the option word of the UFT, then the device inoperable, device offline, and device error bits are set in the IONODE error status word (NODEST), and exit is made through an inline argument. Otherwise, no action is taken and a normal exit is taken.

IS$SEX is a similar routine that processes device under exclusive use conditions. If the I/O call was made with System Recovery specified in the option word of the UFT, then the device under exclusive use and device error bits are set in the IONODE error status word (NODEST), and exit is made through an inline argument. Otherwise no action is taken and a normal exit is taken.

Both routines have identical calling sequences.

Calling Sequence:

```
R3 = Address of IONODE

BLM,R2          IS$SCL . . or (IS$SEX)
DFC             User Error Recovery Module Address
```

Results:

Registers used = R3 (but value not changed).
If extended UFT and System Recovery request, then the IONODE errors status word is updated.

IS$RUF

IS$RUF updates the UFT status word after an I/O operation. The IONODE error status word (NODEST) is copied into the UFT status word with the exception that the buffer busy, the UFT busy bit, and the hold bit are reset. If the operation was a file search, then the device position index is updated in the UFT and in the File Assign Table.

Calling Sequence:

```
R3 = Address of IONODE

BLM,R2          IS$RUF
```

Results:

Registers used = R4, R5, R6, R7, R8, R9, R1Ø, R11
UFT status word updated and device position indexes updated in UFT and File Assign Table, if files search.

IS$XIT

IS$XIT disables the service and data interrupts of a given device. It also resets the controller busy bit in the PDT. Finally, it exits the service interrupt level, thus returning to the interrupted task. (There is no return from this routine.)

Calling Sequence:

```
R1 = Address of Physical Device Table (PDT).

BRU             IS$XIT
```

Results:

The device is disabled and the service interrupt level is exited.

IS$ABO

IS$ABO will cause a task to be aborted from the service interrupt level. The abort is strictly associated with an I/O operation. If the calling task specified local recovery (did not specify System Recovery in the option word of the UFT), it is not aborted, and a normal shutdown must occur after return from this call.

Calling Sequence:

```
R3 = Address of IONODE
R8 = Primary abort argument (CAN-code)
R9 = Secondary abort argument (CAN-code)

BLM,R2          IS$ABO
```

Results:

Registers used = R7, R11

If System Recovery is specified, the TCB is set to abort the task. The address of the abort routine is stored in the "Resume If Interrupted" word of the TCB, the abort arguments are stored in the TCB, and the stall conditions of the status word of the TCB are reset.

IS$REP

IS$REP will will cause the Exceptional Condition task to report any device which fails during an I/O operation. The routine first checks to see if the device that failed is the global CO (Command Output) device. If it is, then the routine exits through the inline argument to try the operation again. Otherwise, the OFL (this node is in progress when offline occurred) bit in the IONODE status word is set, the OL and OM (offline and offline message has not been typed) bits in the status word of the LDT are set, and the Exceptional Condition Task is notified (its suspend bit is reset).

Calling Sequence:

```
R3 = Address of IONODE

BLM,R2          IS$REP
DFC             *ADDRESS OF ROUTINE TO RETRY IF NECESSARY*
```

Results:

Registers used = R4, R5

Exceptional condition task notified of device failure.

IS$TEO, IS$TEF

This routine has two entry points, one for DMP devices - IS$TEO and one for non-DMP devices - IS$TEF.

The routine is called to determine if the next record should be accessed for an advance or backspace operation. An inline argument is taken to end the search in the event of an end of file, end of medium (just for disc files), or device offline condition. Otherwise NODACT (number of records advanced or backspaced in file search) is incremented for advances or decremented for backspaces, and a normal exit is taken.

If the DMP entry, IS$TEO, is used, EOF or EOM is determined from the device status left in R5 in the call. If the non-DMP entry, IS$TEF, is used, EOF is determined by comparing PDTERO with "$$", and if EOF is detected, the EOF bit in the IONODE error status word (NODEST) is set.

Calling Sequence:

The two routines have different calling sequences

```
R1 = Address of PDT
R3 = Address of IONODE

BLM,R2          IS$TEF          * NON-DMP ENTRY
DFC             Routine to Process EOF Termination

R1 = Address of PDT
R3 = Address of IONODE
R5 = Status of device

BLM,R2          IS$TEO          * DMP ENTRY
DFC             Routine to Process EOF Termination
```

Results:

```
Registers used = R5, R6, R7
It is determined whether or not to continue a file search.
```

IS$XSI

IS$XSI is used to prevent a task from accessing a device that is under exclusive use by another task. The routine will also prevent access to a task not conforming to the limit priority, if one was provided at SYSGEN (System Generation). Exception is made to the exclusive use restriction for the OC (Operator Communication) task. If access to a device is denied, exit is made through the inline argument; otherwise, a normal return is made. For a discussion of limit priorities and influence levels, refer to Chapter 1 of the MAX IV General Operating System, System Guide Manual.

Calling Sequence:

```
R1 = Address of PDT
R3 = Address of IONODE

BLM,R4          IS$XSI
DFC             *ADDRESS OF ROUTINE TO PROCESS EXCLUSIVE USE*
```

Results:

```
Registers used = R5, R6, R7, R8, R9, R11, R15
Registers used but not changed = R1, R3
Routines called = IS$TXU
Access rights of a task to a device are checked.
```

IS$DND

IS$DND resolves trailing blank and end of data pointers for byte oriented devices. No change is made to the byte count for non-standard files if the terminate on exact number of bytes bit (TBX) is set in the UFT option word, or if the I/O request was not a READ. Otherwise, twice the number of full words trailing blanks is subtracted from the byte count and the current buffer address word is similarly modified. Next, a null word (#0000) is placed in the buffer (only if the buffer is not full). Finally, the first word of the buffer is checked for "$$" and if found, the EOF bit of the IONODE error status word (NODEST) is set and the new byte count (which has stored in the PDT) is placed in the user's UFT.

Calling Sequence:

```
R1 = Address of PDT
R3 = Address of IONODE

BLM,R2          IS$DND
```

Results:

```
Registers used = R5, R6, R12, R13, R14
Trailing blanks and end of data pointers resolved.
```

IS$AVN

IS$AVN allocates an IONODE from the free node queue. If no node is available, the I/O hold routine IO$IHO is entered until a node becomes free. Once a node is available, it is disconnected from the free node queue, the active node count is incremented by one and the node is marked as out of queue.

Calling Sequence:

R1 = Address of TCB

```
BLM,R7          IS$AVN
```

Results:

Registers used = R4, R5, R6, R8
Address of allocated node in R5.

IS$UNQ

IS$UNQ returns an IONODE to the free node queue. The IONODE to be returned can be the first IONODE in either the slough queue of the Transport Control Table (TCT) or the I/O request queue of the PDT. The IONODE is disconnected from the queue and if the I/O wait service is running, the I/O hold bit in the TCB is reset. A routine is called to return this unqueued IONODE to the free IONODE queue, and the Taskmaster is notified.

Calling Sequence:

R1 = Address of table from which the IONODE is to be unqueued (either the PDT or the TCT).

```
BLM,R2          IS$UNQ
```

Results:

Registers used = R3, R4, R5, R6, R7
Routines called = IS$RTN
An IONODE is returned to the free IONODE queue from either the LDT or the PDT.

IS$STR, IS$SLO

These routines remove an IONODE from the PDT and places it in the slough queue of the TCT. IS$STR will remove the first IONODE in the PDT queue, and IS$SLO will remove any other IONODE in the PDT queue.

Calling Sequence:

These two routines have different calling sequences. The calling sequence for IS$STR (which removes only the first IONODE in the PDT queue) is:

```
R1 = Address of PDT

BLM,R2     IS$STR
```

The calling sequence for IS$SLO (which removes any IONODE from the PDT except the first IONODE) is:

```
R1 = Address of PDT
R3 = Address of IONODE

BLM,R2     IS$SLO
```

Results:

Registers used = R3, R4, R5
An IONODE is removed from the PDT queue and placed in the TCT slough queue.

IS$RVW

IS$RVW removes the first IONODE from the PDT queue and replaces it in the PDT queue according to its priority. This is normally used for file positioning operations and error recovery.

Calling Sequence:

```
R1 = Address of PDT

BLM,R2          IS$RVW
```

Results:

Registers used = R3, R4, R5, R6
The first IONODE in the PDT queue is removed and put back in the PDT queue according to the priority of the task associated with that IONODE.

IS$QUE

IS$QUE moves IONODES from the slough queue of the TCT to the I/O queue of the PDT. All IONODES in the slough queue except those flagged for stall condition (bit KIS in the status word of the IONODE set) are moved. Position of the IONODE in the I/O queue is determined by:

1. If the task requesting the service is the OC (Operator Communication) task, and if it has requested emergency queuing by setting a bit in low core (Bit 15 of LCBSTA), then the IONODE is placed first in the I/O queue.

2. If the device is a terminal, then the position is determined by the following factors:

   (1) the priority of the task making the I/O request
   (2) the I/O operation requested
   (3) whether the device is a read priority or write priority terminal.

3. Otherwise, the position is in order of I/O request within priority (priority of the task making the I/O request).

Calling Sequence:

R1 = Address of LDT

```
BLM,R15          IS$QUE
```

Results:

Registers used = R2, R3, R4, R5, R6, R7, R8, R9
Routines called = IS$TER
All IONODES not in the stall condition are moved from the slough queue of the TCT to the I/O queue of the PDT.

2.4 DATA INTERRUPT LEVEL SYSTEM ROUTINES

ID$DWR, ID$ODA

A call to IS$DWR results in the return of either a carriage control byte or a data byte from the user's output buffer. The return of a carriage control byte is determined by flags and counters in the PDT. One such flag, FBY (First Byte of data), is set in the PDT status word, PDTSTA, if all carriage control functions have been done and data bytes are to be returned for this and subsequent call. ID$DWR "falls through" to ID$ODA to fetch and return a data byte.

ID$ODA is identical to ID$DWR except that the carriage control logic is bypassed.

Calling Sequence:

```
R1 = Address of PDT
R3 = Address of IONODE

BLM,R2     ID$DWR (ID$ODA)
DFC        *ADDRESS OF ROUTINE TO HANDLE TERMINATION*
```

Results:

```
Registers used = R6, R7, R8, R9, R1Ø, R11
R8 = data byte or carriage control byte.
```

ID$DIS

ID$DIS does different things depending on the mode of the operation (Standard, Nonstandard, Binary, ASCII). For all modes except Standard ASCII, a data byte is stored in the I/O buffer by calling ID$DPK (Pack) via an alternate to entry called DIPAK. For Nonstandard ASCII, a check is made for a terminator byte. In the Standard modes, a check is made for the first byte being a binary record header (#Ø3 or #Ø7). If it is, the binary mode is forced, otherwise, the ASCII mode if forced. Logical error and Overflow flags are set in NODEST if a Binary/ASCII mode change occurs. Additionally, the byte count from a binary record is checked and used to update the transfer count.

Calling Sequence:

```
R1 = Address of PDT
R3 = Address of IONODE
R8 = Byte to store in 7-bit ASCII
R9 = Byte to store in 9-bit Binary

BLM,R2          ID$DIS
DFC             (Address of routine to handle terminate)
DFC             (Address of normal return.)
** NO RETURN HERE **
```

Results:

```
Registers used = R6, R7, R8, R1Ø, R11
```

ID$DPK

ID$DPK is called to pack a byte into the input buffer and to keep track of trailing blanks. It calls ID$DNR to terminate the operation if the byte count limit is reached.

Calling Sequence:

```
R1 = Address of PDT
R3 = Address of IONODE
R8 = Byte to be stored in 7-bit ASCII.
R9 = Byte to be stored in 8-bit Binary.

BLM,R2          ID$DPK
DFC             (Address of routine to handle termination)
DFC             (Address of routine on normal return)
** NO RETURN HERE **
```

Results:

```
Registers used = R6, R7, R8, R1Ø, R11
```

ID$DNR

ID$DNR is called when the last byte has been placed in the buffer to reset the Buffer Busy (BB) flags in the UFT and NODEST and to do other end-of-record housekeeping operations.

Calling Sequence:

```
R1 = Address of PDT
R3 = Address of IONODE

BLM,R2          ID$DNR
DFC             (Address of routine to handle termination)
DFC             (Address of routine if termination not
                requested)
**NO RETURN HERE**
```

Results:

Registers used = R8, R10, R11

ID$DTR

ID$DTR is called to terminate an operation and to exit from the DI level. It enables SIs and disables DIs from the group/unit, terminates the group/unit, triggers the Taskmaster and does a CIR from the DI level. It is used for non-multiplexed controllers.

Calling Sequence:

```
R1 = Address of PDT
BRU             ID$DTR
** NO RETURN HERE **
```

Results:

Registers used = R5, R6
Exit from DI level

ID$CXT, ID$CXR

ID$CXT is called to terminate an operation on a multiplexor channel, release the scanner, trigger the Taskmaster, and exit from the DI level. Appropriate commands are issued for 1907 channels.

ID$CXR is similar to ID$CXT except that it just releases the scanner and exits the DI level.

Calling Sequence:

```
R1 = Address of PDT

BRU             ID$CXT (ID$CXR)
** NO RETURN HERE**
```

Results:

```
Exit from DI level
Registers used = R3; R5, R6
```

CHAPTER 3
SYMBIONT TASKS

WARNING

> Since a symbiont is executed in the privileged mode, it cannot be installed into a system and tested without the possibility of a system crash! It should be clear that even though a symbiont is "just a task", it is actually a custom element of the operating system executive and should be treated accordingly by the inexperienced.

3.1 SYMBIONTS

The MAX IV I/O System provides a mechanism that permits the system programmer to interface a task to the Basic I/O System (BIOS) so that it appears to all other tasks as a standard device handler. Such a task is known as a symbiont. As a task, the symbiont may use all of the system services provided for all tasks. It may be resident or non-resident and, in general, execute just like all other tasks.

Symbiont tasks are used to simulate and embody "imaginary" devices with capabilities not usually found in real I/O devices. A symbiont task must be written in Assembly language (no FORTRAN) and must be cataloged as a symbiont and as a privileged task. Symbionts performing certain system functions (that is, spooling) may require being cataloged as unrollable.

Unlike an I/O handler, the symbiont task is never forced to execute code directly because of an interrupt invocation. If there are no operations in the symbiont's PDT (work queue) when an operation is placed there by the I/O system, the symbiont task is activated or resumed, depending on the current state of execution. If, at this time, the symbiont is active and not suspended, no action occurs other than the queuing of the I/O operation node by the calling task and the setting of the symbiont's double resume (RER) bit in TCBOST.

3.1.1 SYMBIONT'S RESPONSE

The symbiont task's responsibility as an I/O system extension is to respond to operation requests placed in its controller table (PDT) by the BIOS on behalf of other tasks in the system. These operations are described by queuing vehicles called nodes and are strung in a doubly-linked list whose head is in the symbiont's PDT. The I/O system places the nodes in this list in order of calling task priority. Refer to Appendix A for av overview of the MAX IV I/O Data Structure Linkage.

To the calling task, the imaginary devices simulated by the symbiont appear as real devices, acting upon what the calling task assumes is an I/O request. The node contains all the information pertinent to the I/O request such as UFT address, buffer address, mapping information, et cetera. The symbiont must eventually respond to the node, processing the I/O request in whatever fashion the symbiont programmer desires.

3.1.2 NODE PROCESSING

When the operation described by a particular node is complete, the node must be removed from the PDT and returned to the list of available nodes. Certain information must be provided to the calling task in its UFT. The most important information is the resetting of the event bits in the UFT status word (UFTSTA) which tell the calling task that the I/O operation it requested is completed. Other information may also be provided such as byte transfer count and device position index information. The symbiont may chose to act on any node in the queue but the normal response is to act only on the first node, often called the current node.

Because of the nature of a multi-programming environment, the symbiont must be prepared to lose an operation node at any time. The node may be lost because the calling task was aborted, or because a TERMINATE request was issued for the current operation. In this case, the I/O system will forceably remove the node from the queue. Whenever this occurs, the I/O system will reset the SD bit in the PDT as a signal that the first node was removed.

The SD bit of the PDT status word (PDTSTA) must be initially set by the symbiont task. Any time the I/O system resets this bit, the symbiont must forget about the first node it was processing and proceed to process the next node which is now the current (first) node or it must process a queue empty condition, whichever is applicable.

If the symbiont was processing other than the first node in the queue, that node can be removed at any time and no notification to the symbiont will be given. However, if bit TER of word PDTSTA had been previously set in the symbiont's PDT, no node will ever be forceably removed from anywhere in the symbiont's queue. Instead, if a TERMINATE request is done from a task abort or a Terminate, the following occurs:

- The symbiont will be resumed.
- The Double Resume bit (RER) in TCBOST of the symbiont TCB will be set.
- The T:RQ (terminate request) bit will be set in PDTSTA of the symbiont's PDT.
- NODUFT of the I/O node being terminated will be set to -1.

The symbiont should periodically examine T:RQ and if set, reset T:RQ and then search its queue for I/O nodes with NODUFT set to minus 1 (-1). When such a node is found, the symbiont should remove that node as quickly as possible since the task causing the termination of that node will not be able to abort (if aborting) or otherwise use that node (if terminating) or the UFT until the symbiont removes the node.

3.1.3 CONTROLLER BUSY BIT

As was mentioned earlier, the symbiont may be resumed (if suspended) when an I/O operation is placed in its queue. This is provided through another bit in the PDT called controller busy (CB). When the I/O system places an operation in the queue, it checks CB and if set, takes no further action other than setting the RER bit. If CB is reset, the I/O system first sets it, then checks to see if the symbiont is active. If so, the symbiont is unsuspended and a context switch is requested. If the symbiont was not suspended, the unsuspend action has no effect. In any case, the context switch will cause the symbiont to run again if it was relinquishing.

Ultimately, this bit is used by the symbiont any time it wishes to be notified when an operation is placed in its queue (by resetting CB, it will then be set when a new node is put in the queue). This notification applies even if the symbiont were to exit after resetting CB.

3.1.4 I/O OPERATION COMPLETES

There is another bit available to the symbiont or for any task that will cause the symbiont to be notified when any I/O operation it is performing completes. This bit is R:IO in TCBOST of the symbiont's TCB.

When this bit is set and any I/O operation queued by the symbiont completes, the symbiont will be resumed (if suspended) and the RER bit will be set. The symbiont's I/O operations must be Quick Return if the use of R:IO is to be meaningful.

If the symbiont sets R:IO, the symbiont need test and reset only RER to find out if either a new node was queued to the symbiont OR an I/O operation done by the symbiont has completed. R:IO may be set by the PECULIAR RIOCOMPLETE directive in TOC. Refer to the MAX IV TASK OVERLAY CATALOGER, Programmer's Reference Manual.

3.1.5 EXCLUSIVE USE AND INFLUENCE LEVEL

All standard "real" I/O device handlers provide privileged tasks the capability to take exclusive use of a device. System generation provides a means to define a device as having permanent exclusive use taken by a selected task.

In addition, an influence level may be specified for a device limiting access to only tasks with sufficient "political" power. This influence level is checked against the influence level of each task queuing operations to the device. If the task has a lower level (higher number) than the level of the device the operation is prohibited.

A symbiont task may also perform the exclusive use and influence level function of a device in order to appear to the system just like all handlers. The exclusive use request by another task causes the node to be stripped (placed in a special "slough" queue in the TCT) for another try later.

If a device has permanent exclusive use assigned or if the influence level is incompatible, the result depends on the recovery mode of the calling task. If the task requested local recovery, the ER and INO bits are set in the UFT status word (UFTSTA) and the operation is normally shutdown (unqueued) without processing of the node. If the calling task specified System Recovery, it is normally aborted (KILL service). Remember, a symbiont may take any action desired, including ignoring the condition; this is just what the I/O handlers do.

3.1.6 SOFTWARE OFFLINE

Another standard feature of the I/O system is the "software" offline feature. At any time, the operator can put a device in the offline state. When this happens, the BIOS will not queue operations to the device. All operations already queued are stripped from the queue by the device handler. This can be inhibited at system generation.

Anytime a real device has an unrecoverable error and System Recovery was specified by the current operation, the device is put offline by the handler. In this case the Exceptional Condition task (X) is invoked to report the error and all operations in the queue are stripped. A symbiont may use the offline feature just as the I/O handlers do or it may choose to ignore the situation altogether and allow all queued operations to run regardless of the offline state.

Remember, though, that if the operator puts the imaginary device offline, the BIOS may not queue any more operations to the symbiont until the offline state is removed. If the symbiont choses to strip nodes from the queue when offline is detected, they will be queued by BIOS when the device is put back online by the operator.

3.1.7 ADDITIONAL CAPABILITY

For additional request handling capability symbionts may make use of the optional task-level and test assign symbiont PDT vectors. The suffix of these optional routine names are supplied on the SYMCONTROL statement in SYSGEN.

The task-level routine is invoked by a direct branch from BIOS. Return is usually to IO$IOR to enqueue and start the I/O operation.

The test assign routine is invoked by a branch and link from IO.TAS. The entry registers are as follows:

```
R2   ADDRESS OF BIOS UFT ON STACK
R3   ADDRESS OF LDT
R4   ADDRESS MINUS 1 OF MAIN STACK REGISTERS
R5   ADDRESS OF PDT
R6   ADDRESS OF TCT
R7   SUBROUTINE LINKAGE
R8   ORIGINAL CONTENTS OF USER'S R8
```

Return either to the address in register 7 or to IO$XIO to either continue the test assign call as is or to process the call yourself and exit the I/O request.

3.2 MAPPING REQUIREMENTS

The average symbiont must contend with three mapping spaces in order to carry out its I/O related functions. These are:

1. Its own single map addressing space
2. MAP Ø addressing space
3. The operand addressing space of the task that queued the current operation.

If the symbiont is to be made permanently resident, MAP Ø will be its map. A symbiont may be written to execute in either MAP Ø or a private map without knowledge of its residency.

All the symbiont data structures provided to it by BIOS (PDT, Nodes, LDT and other I/O related data structures) are located in MAP Ø. Therefore, the symbiont must Select MAP Ø (SZOM) before accessing these data structures. Obviously, if the symbiont then wishes to access operands (data) in its own body, it must reselect its own map. The procedure to do this will be discussed later. If the symbiont wishes to access data in the body of the calling task (I/O buffers, UFT, et cetera) it must use special instructions to do so. This is the main reason that a symbiont task must be privileged.

3.2.1 SELECTING MAP Ø

The symbiont has no problem getting into MAP Ø. It simply executes:

```
     SZOM                    ...select MAP Ø as my operand addressing
*                              space
```

Since MAP Ø is always active, this instruction may be executed at any time it is desired. Once selected, the Symbiont's PDT address will probably want to be obtained. The following sequence is used.

```
     LDM,R5    LCBTCB        ...get symbiont's TCB address
     LDM,R6,R5 TCBPDT        ...get address of PDT from TCB
```

The Symbiont PDT address is now in R6. Since the Symbiont is active when it executes these instructions, LCBTCB always points to its TCB. All symbionts have these PDT pointers in their TCB. (The word index of any symbols used can be found in the descriptions of the data structures.)

From this point, all pertinent information can be obtained from two tables, the TCB and the PDT. The PDT contains the pointers to the start and end of the node queue and each node contains information and address pointers to all other necessary data structures. Refer to the MAX IV DATA STRUCTURES, System Design Manual.

3.2.2 SELECTING THE SYMBIONT'S MAP

When it is desired to return to the Symbiont's map, the required sequence is: (assume R5 still contains the TCB address).

```
     SIA,#F                  ...lock out Taskmaster (do not let
*                              maps change)
     LDM,R7,R5  TCBRES       ...get task's assigned maps from TCB
     SOOM,R7                 ...select its operand map (OM) as the
*                              current OM
     RIA,#F                  ...it is safe to let the task's maps
*                              change now
```

While a task is executing, any time it is interrupted, its currently active map may change. TCBRES always contains the resources currently assigned to a task. The symbiont must "lock-out" the Taskmaster so that it can't be interrupted between the next two instructions. Obviously, interruption could cause the resources to change which would consequently change TCBRES. When the symbiont is re-entered, R7 would be obsolete.

Map selection based on old information could possibly cause a system halt. Therefore, the lockout prevents any resource change so the map selected will be the map assigned. Once selected, the lock-out may be removed. From this point, interruption does not matter since the current PS will be changed along with TCBRES.

3.2.3 ACCESSING INFORMATION IN THE CALLING TASK

The Symbiont executes asynchronously with respect to the calling task. Therefore, there is never a guarantee that the calling task will have an active map when the symbiont needs to access the calling task's addressing space. Indeed, the symbiont may be the culprit who stole the map!

There is, however, a map image associated with the addressing space of every task in the system. Using the map image is the only reliable means of data transfer between the symbiont and its calling task. Methods for using the map image for this purpose are described later.

3.3 PRIORITY OF THE SYMBIONT

The symbiont normally runs at a higher priority than any task accessing it. This is only required if the TER bit of PDTSTA was not set by the symbiont. The TER bit will keep a task that has I/O nodes queued to the symbiont from leaving the system while the symbiont is accessing the calling task.

- If the TER bit is set, the symbiont may do I/O directly into the caller's memory and be at a lower priority than the caller.
- If the TER bit was not set, the symbiont cannot do I/O into the caller's memory and must be higher in priority than the caller.
- If the caller is higher in priority and TER is not set, the caller could leave the system and another task could use the same TCB the original caller left behind all before the symbiont runs again, the symbiont will then be accessing the wrong task's Map Image, possibly causing a system halt.

3.4 THE I/O BUFFER

The node provides the mapping information about the I/O buffer of the calling task. The Map Image Actual Page (MIAP) in the node (NODMAP) specifies the map to be used for the data transfer. The map length (NODLEN) is provided so that the symbiont can check to be sure the buffer resides entirely within the addressing space defined. This is necessary because a map image may not be 256 words long, and other information may exist beyond the map image.

The mapping hardware will assume this data is actual page definitions if they are referenced. To access the first word of the I/O buffer the following sequence might be used: (Assume R7 contains the node address)

```
     LDM,R5,R7  NODMAP    ...get map image actual page address
     LDM,R4,R7  NODBUF    ...get virtual address of 1st (Øth) word
*                           of buffer
     LDVM,R8,R4           ...get 1st word via direct map image
*                           translation
     HOS,xxxxxx           ...HOP if word not in allocated memory
```

If subsequent data words are desired, incrementing R4 (in the above example) by one each time will do the job. Obviously, the data access must stop when the end of the buffer is reached as specified by the byte count word (NODCNT). Note this word is a byte count. The LDVM instruction loads a word at a time. Thus, the counter must be incremented by two for each iteration.

If data is to be stored in the calling task addressing space, the LDVM should be replaced by an STVM.

3.5 THE UFT

The UFT must be accessed generally by the same method as the I/O buffer, except it may be in a different addressing space. The UFT may be addressed through the operand addressing space of the calling task or through MAP Ø. (The I/O buffer may be in another map altogether.) Bit MPØ of the node status word (NODSTA), when set, indicates the UFT is addressed through MAP Ø addressing space. When reset, it indicates the operand addressing space of the calling task. The LDVM and STVM instructions can be used regardless of the UFT mapping since we know that MAP Ø MIAP is actual page 1. For example, consider that we wish to clear the UFT status word: (R7=node address).

```
     LBR,R5,B15             ...set MAP Ø MIAP address (page 1)
     TBXB,MPØ,R7  XYZ       ...test for MAP Ø, branch if true to XYZ
     LDM,R6,R7    NODTCB    ...no, get calling task's TCB address
     LDM,R5,R6    TCBAAO    ...and fetch its operand map image proper
*                             addressing space now selected
XYZ  LDM,R4,R7    NODUFT    ...get calling task's UFT address
     ZRR,R8                 ...
     STVM,R8,R4             ...store zero in first word of UFT
```

At first we assume the UFT is in MAP Ø above, so we set actual page 1 as the MIAP of the UFT. If bit MPØ of the node status word is set, we were right and the map image is used. If Bit MPØ is reset, we must determine the MIAP of the task's operand map. The node contains the TCB Address (NODTCB). Word TCBAAO contains the operand MIAP (Map Image Actual Page). We now simply load that information. At this point, the MIAP has been found. It is now a simple matter to load the UFT address and store zero in word zero. Obviously we would not want to go through this sequence more than once, so it is desirable to save the MIAP for subsequent use.

3.6 SUBROUTINES AVAILABLE TO THE SYMBIONT

Now that we have covered the details of elemental symbiont functions, the programmer really does not have to worry about most of the problems. Subroutines are provided in the MAX IV System Element Library to aid the symbiont programmer in system data structure manipulation.

With each revision of MAX IV, any system data structure may change. If the interface subroutines provided are used to access these structures, the programmer need not be concerned with the changes. If information is required that is not provided by the subroutines, the programmer must examine any data structures accessed when a new version of the OS is received.

The subroutines use registers 4 through 1Ø except where noted. When the subroutines return to the symbiont, the condition codes may be tested to determine the success of the requested function. If compatible MAX III symbionts are to be written by the programmer, register 1Ø may be tested for condition codes where bits 12 through 15 correspond to condition codes NZOC, respectively. Although MAP Ø is selected by these routines, the symbiont's map is reselected before return. These symbiont subroutines are re-entrant.

As of Revision H.Ø of MAX IV the symbiont subroutines contain code relocation counters. Counter 2 is used for instructions, and counter 4 for constant operands. As of Revision H.Ø, the variable portion of the symbiont subroutines were moved to the TCB extension of each symbiont task (EXTSSV, EXTSTA). If the user-written symbiont also contains these counters and is implemented in a sharable manner, then the symbiont can be linkedited using the MAX IV Link Editor and the SLM directive to produce a sharable symbiont.

With the addition of the code relocation counters, the symbiont service routines are now also usable in a two map-task environment.

If the symbiont is a MAP Ø resident task, the symbiont service routines operands are sharable and re-entrant.

SS$SIO

This routine should be called at the start of each operation. The subroutine will initialize the PDT of the symbiont for use by other subroutines that may be called. Additionally, SD (shutdown) is set so the loss of a node can be detected. The MIAP (Map Image Actual Page) of the UFT is determined and stored in word PDTUAP. If the I/O request was READ or WRITE, the specified transfer count is limited to the device limit set at system generation and the buffer is checked to insure it resides within the map image.

Calling Sequence:

```
EXT       SS$SIO    ...needed only once in the program
BLM,R8    SS$SIO    ...call "setup I/O" subroutine
```

Results:

Registers used = R5 through R10
R6 = address of first node in queue
R9 = REX Call Number

Condition Codes Returned:

NZOC
1000 - PDT is successfully initialized.
0000 - The PDT is initialized, but the device was logically offline.
0001 - The queue is empty (no nodes).
0010 - The I/O buffer is outside the map image addressing space. The PDT is initialized but any data transfer will yield unpredictable results.

SS$TXU

This subroutine will compare the exclusive use status and influence level of the device against the task whose I/O request is at the top of the queue. This routine should be called after SS$SIO.

Calling Sequence:

```
EXT       SS$TXU     ...needed only once in the program
BLM,R8    SS$TXU     ...is everything legal?
```

Results:

Registers used = R4 through R1Ø

Condition Codes Returned:

NZOC
1ØØØ - The device is not under exclusive use or the task requesting the I/O operation has the device. The limit level is also within range.
ØØØØ - The node has been removed from the queue (SD is reset).
ØØØ1 - The device is temporarily under exclusive use by a task other than the task whose I/O request is current.
ØØ1Ø - The device is under permanent exclusive use by a task other than the task whose I/O request is current or the limit level of the task is lower than that of the device.

SS$STR

This subroutine will strip the node from the top of the PDT and place it in the slough queue of the Transport Control Table. This is usually done for offline or temporary exclusive use conditions such as when a node cannot be processed immediately and there are other nodes in the queue.

Calling sequence:

```
EXT       SS$STR     ...needed only once in the program
BLM,R8    SS$STR     ...strip node from main I/O queue
```

Results:

Registers used = R5 through R1Ø

Condition Codes Returned:

NZOC
1ØØØ - The node has been stripped.
ØØØØ - The node was removed before this routine processed it (SD is reset).

SS$GTW

This subroutine will retrieve data words from the I/O buffer of the task whose operation is current. Each call to this routine will load the next sequential word in the buffer. End of buffer detection is provided. Before this routine is used, SS$SIO must have been called. This is because the buffer pointer and count are maintained in the PDT and are initialized by SS$SIO. This routine must not be called unless WRITE is the I/O request. If the symbiont wishes to support the defined data transmission modes, it is the programmer's responsibility to test for null words, carriage returns, and UFT supplied terminators.

Calling Sequence:

```
EXT       SS$GTW    ...needed only once in the program
BLM,R8    SS$GTW    ...get data word
```

Results:

Registers used = R4 through R10
R9 = Data word retrieved from I/O buffer.

Condition Codes Returned:

NZOC
1000 - Data in R9 is valid.
0000 - The node has been removed from the queue. R9 is not valid data (SD is reset).
0001 - Device is logically offline. R9 still contains valid data.
0010 - The end of the buffer has been reached. The transfer count has expired. The contents of register 9 is not valid data.
0100 - Only the left byte of R9 is valid. The last data word was accessed and the transfer count is odd.

More than one of the above condition codes may be set if multiple conditions are found; for example, the device is logically offline and the end of the buffer is reached (CC=0011).

SS$PTW

This routine will store data words into the I/O buffer of the task whose operation is current. Each call to this routine will store the desired data into the next sequential word in the buffer. End of buffer detection is provided. Before this routine is used, SS$SIO must have been called. This routine must not be called unless READ is the I/O request.

Calling Sequence:

Register 9 must contain the data word to be stored.

```
EXT       SS$PTW    ...needed only once in the program
BLM,R8    SS$PTW    ...put data word
```

Results:

Registers used = R4 through R10

Condition Codes Returned:

NZOC

1000 - the data word was stored

0000 - The node has been removed from the queue and the data was not stored (SD is reset).

0001 - The data was stored but the device is logically offline.

0010 - The end of buffer has been reached and the data is not stored.

More than one of the above condition codes may be set if multiple conditions are found; for example, the device is logically offline and the end of the buffer is reached (CC=0011).

SS$UNQ

This subroutine will remove the current node and return it to the node pool of the task associated with the I/O request. The UFT busy condition is reset and the desired status is stored for examination by the calling task. The byte transfer count word in the UFT (UFTBCT) is also stored with the specified information. This routine is called by the symbiont when an operation is completed, or at any time the symbiont decides the current operation can no longer be processed and must be shutdown.

Calling Sequence:

Register 9 contains the desired status to be stored in the UFT status word (UFTSTA). Bits 0-8 and 12-15 should be zero to indicate a fully successful operation. Bits 9 through 11 may be set to indicate event conditions.

Register 10 contains the information to be placed in the UFT byte transfer count word (UFTBCT). This should be the accumulated count of the number of bytes processed by the symbiont for READ or WRITE and zero for all other I/O requests.

```
EXT       SS$UNQ     ...needed only once in the program
BLM,R8    SS$UNQ     ...terminate the caller's I/O operation
```

Results:

Registers used = R4 through R10

Condition Codes Returned:

NZOC

1000 - The node has been removed from the queue and placed in the free node pool of the task.

0000 - The node was already removed from the queue (SD reset).

SS$RVW

This subroutine will remove the current node and place it in the symbiont's PDT I/O node queue according to its priority. This is used in the case that the symbiont wishes to process this node later if higher priority operations could be processed first. Normally all nodes are queued behind the first node in an order determined by the priority of their calling task. If a node is queued which is higher in priority than the first node, it is still queued behind it. The symbiont may find a delay condition or break point which could allow processing of higher priority nodes, as in simulating long operations such as file searches, which require many iterative READ operations. If the routine is called and the current node is already the highest priority node, no action occurs. In any case, the symbiont must be prepared to work on another node (or no node at all) after calling this subroutine.

Calling Sequence:

```
EXT       SS$RVW    ...needed only once in the program
BLM,R8    SS$RVW    ...is top node still highest priority?
```

Results:

Registers used = R4 through R1Ø

Condition Codes Returned:

```
NZOC
1ØØØ - The node has been requeued.  It may still be the
       first (current) node.
ØØØØ - The node is no longer in the queue (SD reset).
```

SS$QUE

This subroutine will remove all I/O nodes from the TCT Queue that are not flagged as KEEP IN SLOUGH QUEUE (KIS) and place them in the PDT Queue. The nodes will be placed in order of priority and nodes of like priority will be placed in order of their request.

Calling Sequence:

Register 4 contains the LDT address.

```
EXT       SS$QUE    ...needed only once in the program
BLM,R8    SS$QUE    ...queue sloughed I/O nodes from TCT to PDT
```

Results:

Registers used = R4 through R1Ø

Condition Codes not used.

SS$QUA

This is an alternate entry point into the SS$QUE subroutine. This entry point does not require register 4 to be present with the LDT address.

Calling Sequence:

```
     EXT      SS$QUA    ...needed only once in the program
     BLM,R8   SS$QUA    ...queue sloughed I/O nodes from TCT to PDT
```

Results:

Registers used = R4 through R1Ø

Condition Codes not used.

SS$DEL

This subroutine is called when the symbiont wishes to enter the delay service; for example, when it has no more nodes to process. Due to the multi-programming environment, a node could be queued while the symbiont is in the process of entering the delay service but has not actually begun its delay. If the symbiont wishes to delay, but wishes to be unsuspended early if a node should be queued, this routine will provide the necessary sequence to accomplish this. If bit CB of word PDTSTA is set when this service is entered, it will return without delay. An entry point is provided to force the delay even if CB is set. In this case, CB will be reset before the delay is entered. If the symbiont wishes to delay and does not wish to be resumed early, it can simply call the delay service directly.

Calling Sequence:

Registers 14 and 15 must contain the standard delay service arguments for the MAX IV REX call.

```
     EXT      SS$DEL    ...needed only once in the program
     BLM,R8   SS$DEL    ...suspend, but inform me if more nodes
*                         are queued
```

Results:

Registers used = R5 through R8, R1Ø

Condition Codes Returned:

```
     NZOC
     1ØØØ - Returned from suspend.
     ØØØØ - Suspend never entered because CB was set before
            the call.
```

SS$DEF

This is an entry point to SS$DEL and will cause CB to be reset before entering the delay service. This is useful if a delay is desired even if nodes are in the queue but an early resumption is wanted when any new node is queued. The calling sequence is equivalent to SS$DEL except that a condition code of 0000 will not be returned.

Results:

Registers used = R5 through R8, R10

SS$SUS

This subroutine, like SS$DEL, provides the sequence necessary to suspend the symbiont without missing a queued node. The only difference is that this service will never return control until some node is queued.

Calling sequence:

```
EXT      SS$SUS     ...needed only once in the program
BLM,R8   SS$SUS     ...suspend me until a node is queued
```

Results:

Registers used = R5 through R8, R10

Condition Codes Returned:

NZOC
1000 - Returned from suspend.
0000 - Suspend never entered because CB was set before the call.

SS$SUF

This is an entry point to SS$SUS and will cause CB to be reset before suspending. This is useful if a suspend is desired even if nodes are in the queue, but a resumption is wanted when any new node is queued. The calling sequence is equivalent to SS$SUS except that a condition code of 0000 will not be returned.

Results:

Registers used = R5 through R8, R10.

SS$EXI

This subroutine, like SS$DEL, provides the sequence necessary to exit the symbiont without missing a queued node.

Calling Sequence:

```
EXT      SS$EXI    ...needed only once in the program
BLM,R8   SS$EXI    ...exit, restart me if node is ever queued
```

Results:

Registers used = R5 through R8, R10

Condition codes are not returned.

If a return occurs from this subroutine the EXIT service was not entered. Otherwise, the symbiont will be activated when a node is queued and execution will begin at its entry point.

SS$EXF

This is an entry point to SS$EXI and will cause the symbiont to unconditionally exit. CB is reset before the exit is called, so a new node queued will cause reactivation.

Calling Sequence:

```
EXT      SS$EXF    ...needed only once in the program
BRU      SS$EXF    ...exit unconditionally, restart on new node
```

The EXIT service should never be called while CB is set, because new nodes being put in the queue will not cause reactivation.

Results:

Registers used = R5 through R8, R10

Condition Codes not used.

SS$GPI

This subroutine will retrieve the current File Position Index (FPI) of the logical file through which the current node was queued. The FPI will be loaded by the system from the FAT or the UFT depending on the state of the random bit (RAN) in the option word (NODOPT). The FPI will be returned to the symbiont in a 32-bit format even though it could be retained by the calling task in 16-bit format. This subroutine is normally called for the purpose of FPI updating and is intended to be used in conjunction with SS$PPI. When simulating "randomly addressable" devices, the symbiont may need to fetch the FPI prior to starting an operation.

Calling Sequence:

```
EXT       SS$GPI     ...needed only once in the program
BLM,R8    SS$GPI     ...get logical file's of UFT's currrent FPI
```

Results:

Registers used = R4 through R10
Registers 6 and 7 = the FPI as a 32-bit integer
(R6 = MSH, R7 = LSH)

Condition Codes Returned:

NZOC
1000 - The FPI is returned in R6, R7.
0000 - The node has been removed from the queue and the FPI is not returned (SD is reset).
0001 - The FPI is returned, but the device is logically offline.

SS$PPI

This subroutine will store the 32-bit FPI back into the UFT of the task whose I/O request is current. The FPI will be stored in 32-bit format in the FAT and 16- or 32-bit format in the UFT. If the UFT specifies 16-bit format, the most significant 16-bits will be lost.

Calling Sequence:

Registers 6 and 7 contain the FPI as a 32-bit integer.
(R6 = MSH, R7 = LSH)

```
EXT      SS$PPI     ...needed only once in the program
BLM,R8   SS$PPI     ...put FPI into logical file
```

Results:

Registers used = R4 through R10

Condition Codes Returned:

NZOC
1000 - FPI has been stored.
0000 - The node has been removed from the queue and the FPI was not stored (SD reset).
0001 - FPI was stored but the device is logically offline.

SS$KIL

This routine is called to cause the task whose node is current to be aborted. When return from this routine occurs, the node may still be in the queue. This is because the abort will occur at the priority level of the aborting task. A subsequent call to SS$UNQ will insure removal of the node.

Calling Sequence:

Register 13 = the 3-character CAN-code reason for the abort.

```
EXT      SS$KIL    ...needed only once in the program
BLM,R8   SS$KIL    ...kill the task who gave me this node
```

Results:

Registers used = R4 through R8, R10, R14, R15
Register 14 and 15 = the 1- to 6-character task name in CAN-code format.

Condition Codes Returned:

NZOC
1000 - Task is aborting.
0000 - The node has been removed from the queue. The task has not been aborted by this call.
0001 - The task is aborting but the device is logically offline.
0010 - Task not in CPUQ or incompatible.

SS$REP

This subroutine will put a device logically offline and cause the condition to be reported to the operator. The device selected will be that device corresponding to the first node in the queue. If the device is already offline , the condition will not be reported. This routine is usually called after the symbiont wishes to inform the calling task of an unrecoverable error that has transpired which is related to the I/O operation being simulated and the calling task has requested the System Error Recovery mode in its UFT. The symbiont normally "sloughs" the node in the logical slough queue and, if it can process other nodes for other logical devices it is simulating it proceeds.

Calling Sequence:

```
EXT       SS$REP     ...needed only once in the program
BLM,R8    SS$REP     ...inform operator of bad device status
```

Results:

Registers used = R5 through R8, R10

Condition Codes Returned:

NZOC

1000 - Device has been put offline and the condition is being reported.

0000 - The node has been removed from the queue. The device is not put offline (SD is reset).

0001 - The device is already offline. The condition is not reported.

SS$LDT

This subroutine will return some of the data items from the device table to the symbiont. This information includes parameters about the logical device that the symbiont is simulating. Since the information is contained in system data structures in MAP Ø, these items may change from one release to another. If the programmer uses the subroutine, those changes will be transparent. Additionally, if the programmer is coding a symbiont to run under MAX III and IV, its use is obvious.

Calling Sequence:

```
     EXT      SS$LDT     ...needed only once in the program
     BLM,R8   SS$LDT     ...get information from operation's
*                           logical device
```

Results:

Registers used = R5 through R8, RlØ
R5 = the contents of Word Ø (LDTSTA) - device status.
R6 = the contents of Word 1 (LDTNAM) - device name.
R7 = the contents of Word 3 (LDTTYP) - device classification and configuration bits.

Condition Codes Returned:

NZOC
1ØØØ - Information is in registers.
ØØØØ - The node has been removed from the queue. The information requested is not in the registers (SD reset).
ØØØ1 - The information is returned but the device is logically offline.

SS$LDA

This is an entry point to SS$LDT and is called to return information from other than the current node's LDT. In this case the programmer wishes to get LDT information from a node that is not at the head of the symbiont's PDT and does not want to move it to the head.

Calling Sequence:

Register 9 contains the address of the node from which the LDT address is to be retrieved to pass the information.

```
EXT       SS$LDA     ...needed only once in the program
BLM,R8    SS$LDA     ...get information from specified LDT
```

Results:

Registers used = R5 through R8, R10
R5, R6, and R7 = same as SS$LDT

Condition Codes Returned:

Same as SS$LDT except 0000 will not be returned.

SS$NOD

This subroutine will return some of the data items from the node to the symbiont.

Calling Sequence:

```
EXT       SS$NOD     ...needed only once in the program
BLM,R8    SS$NOD     ...get information from the current node
```

Results:

R2 = the contents of Word 0 (NODSTA) - node status.
R3 = the contents of Word 4 (NODOPT) - node and UFT options.
R4 = the contents of Word 9 (NODBUF) - I/O buffer or chain.
R5 = the contents of Word 12 (NODMAP) - transfer MIAP.
R6 = the contents of Word 11 (NODLEN) - MIAP length.
R7 = the contents of Word 10 (NODCNT) - transfer count.

Condition Codes Returned:

NZOC
1000 - The information is in the registers.
0000 - The node has been removed from the queue. The information request is not in the registers (SD reset).
0001 - The information requested is returned but the device is logically offline.

SS$NDA

This is an entry point to SS$NOD and is called when information is to be returned from a node other than the one at the head of the symbiont's PDT.

Calling Sequence:

Register 9 contains the address of the node from which the information is desired.

```
EXT       SS$NDA     ...needed only once in the program
BLM,R8    SS$NDA     ...get information from the specified node
```

Results:

Registers used = R2 through R8, R10
R3 through R7 = same as SS$NOD

Condition Codes Returned:

Same as SS$NOD except condition code 0000 will not be returned.

SS$NOE

This subroutine will return the same data items from the node as SS$NOD does. It also returns two additional items.

Calling Sequence:

```
EXT       SS$NOE     ...needed only once in the program
BLM,R8    SS$NOE     ...get information from the current node
```

Results:

Registers used = R2 through R8, R10, R14, R15
R2 through R7 = same as SS$NOD
R14 = the contents of Word 18 (NODHU1) - handler use word 1.
R15 = the contents of Word 19 (NODHU2) - handler use word 2.

Condition Codes Returned:

NZOC
1000 - The information is in the registers.
0000 - The node has been removed from the queue. The information request is not in the registers (SD reset).
0001 - The information requested is returned but the device is logically offline.

SS$NAX

This is an entry point to SS$NOE and is called when information is to be returned from a node other than the one at the head of the symbiont's PDT.

Calling Sequence:

Register 9 contains the address of the node from which the information is desired.

```
EXT      SS$NAX     ...needed only once in the program
BLM,R8   SS$NAX     ...get information from the specified node
```

Results:

Registers used = R2 through R8, R10, R14, R15
R2 through R7 = same as SS$NOD
R14 and R15 = same as SS$NOE

Conditions Codes Returned:

Same as SS$NOE except 0000 will not be returned.

SS$NXT

This subroutine will return the address of the next node in a queue. The programmer passes the address of any queued node to the subroutine and it will pass the address of the next one.

Calling Sequence:

Register 9 contains the address of a queued node.

```
EXT      SS$NXT     ...needed only once in the program
BLM,R8   SS$NXT     ...get me the address of the next node
```

Results:

Registers used = R5 through R10
R9 = the address of the next node

Condition Codes Returned:

NZOC
1000 - Register 9 is valid.
0001 - The node specified at entry is the last node in the queue. Register 9 is unchanged.

SS$NMV

This subroutine will move the designated node to the head of the symbiont's PDT. The node specified must be queued somewhere in the PDT queue. It may already be at the head.

Calling sequence:

Register 9 contains the address of the node to be removed.

```
EXT       SS$NMV     ...needed only once in the program
BLM,R8    SS$NMV     ...put this node at the head of my PDT
```

Results:

Registers used = R4 through R8, R1Ø

Condition Codes Returned:

```
NZOC
1ØØØ - The node is at the head of the PDT.
ØØØ1 - The node was already at the head of the queue.
```

SS$TAK

This subroutine is provided to enable the symbiont to respond to a TAKE exclusive use request. It is called only if the symbiont programmer wishes to support the feature, but is necessary if the imaginary devices to be simulated are to appear as standard "MAX IV-like devices" to their calling tasks. If this call is desired, it will be indicated by a call number of 9 even though the actual number is #23. The BIOS makes the conversion for processing convenience at the handler and symbiont level.

Calling Sequence:

```
EXT       SS$TAK     ...needed only once in the program
BLM,R8    SS$TAK     ...process take on behalf of calling task
```

Results:

Registers used = R4 through R1Ø

Condition Codes Returned:

```
NZOC
1ØØØ - Exclusive use is taken.
ØØØØ - Node has been removed (exclusive use not taken).
ØØØ1 - Exclusive use was taken but the device is
       logically offline.
```

SS$GIV

This is an entry point to SS$TAK and is designed to respond to a GIVE exclusive use request. It will cause the exclusive use words in the logical device table to be cleared. This request is indicated by a call number of 10 even though the actual number is #24. The BIOS makes the conversion for processing convenience. The calling sequence is equivalent to SS$TAK.

Results:

Registers used = R4 through R10

Condition Codes not used.

SS$GBF

This subroutine will move the contents of the I/O buffer in the calling task's addressing space to a local buffer within the symbiont's addressing space.

Calling Sequence:

Register 14 contains the address of the symbiont's local buffer.

```
EXT      SS$GBF    ...needed only once in the program
BLM,R8   SS$GBF    ...move the user's I/O buffer to mine
```

Results:

Registers used = R4 through R10, R14

Condition Codes Returned:

NZOC
1000 - The contents of the buffer have been moved.
0000 - The node is no longer at the head of the PDT (SD reset). The symbiont's local buffer is unchanged (the data is not moved).
0001 - The device is logically offline but the operation occurred.

SS$SBF

This is an entry point in SS$GBF and is called to move the contents of the local symbiont's buffer into the I/O buffer in the calling task's addressing space.

Calling Sequence:

Register 14 contains the address of the symbiont's local buffer.

```
EXT      SS$SBF    ...needed only once in the program
BLM,R8   SS$SBF    ...move my buffer to the user's I/O buffer
```

Results:

Registers used = R4 through R10, R14

Condition Codes not used.

SS$TCB

This subroutine will return the task name for the TCB of the requesting task.

Calling Sequence:

Register 9 contains the address of the TCB from which the information is desired.

```
EXT      SS$TCB    ...needed only once in the program
BLM,R8   SS$TCB    ...get TCB information
```

Results:

Registers used = R2 through R8, R10
R2 = the contents of Word 39 (TCBNAM) - task name
R3 = the contents of Word 40 (TCBNAX) - task name extension

Condition Codes Returned:

NZOC
1000 - The information is in the registers.
0000 - The node has been removed from the queue. The information request is not in the registers (SD reset).
0001 - The information requested is returned but the device is logically offline.

SS$TCA

This is an entry point to SS$TCB and is called when a task name is to be returned for a TCB other than the current node TCB.

Calling Sequence:

Register 9 contains the address of the TCB from which the information is desired.

```
EXT      SS$TCA    ...needed only once in the program
BLM,R8   SS$TCA    ...get me information
```

Results:

Registers used = R2 through R8, R1Ø
R2 and R3 = same as SS$TCB

Condition Codes Returned, only N is set.

3.7 EXAMPLE SYMBIONT NARRATIVE

The following description of a sample symbiont is provided to aid the system programmer in the implementation of a symbiont. The name of the symbiont is COMPRS and is designed to take data from the calling task's buffer and compress it into a buffer local to COMPRS. When the compression buffer is full COMPRS will write it to a real device. The details of the actual compression and subsequent output to the real device are not germane to the example and will not be included in the description.

The section of an "output" spooler which moves data queued to imaginary devices from their calling task to secondary storage is a typical application of the symbiont described in this example. It should be noted that when this symbiont has left over time (that is, when there are no nodes for it to process), it can do other useful things such as disposing of data already placed on secondary storage to some real output device. The rest is left to the system programmer's imagination.

When the symbiont is first activated, it calls SS$SUS so that it will suspend if it was activated by the operator or some mechanism other than the queuing of a node. If a node was queued causing activation, the suspend will not occur. Note that SS$DEL could be called if the symbiont wished to limit the time it waits for someone to queue an operation node to it.

Assume a node is in the queue. SS$SIO is called for every new node to initialize the PDT. On return, the condition codes are tested to determine the following:

1. Queue empty? - Suspend

2. Device offline? - Strip node and go back to process next node.

3. Buffer not contained entirely in Map? - abort calling task if System Recovery specified, otherwise, return node with error.

If the condition codes indicate everything is normal, the symbiont proceeds on its normal course. The logical device is checked for exclusive use and influence level restrictions. If the device is under permanent exclusive use or the influence level does not allow the task to access it, the condition is considered a fatal error and the same course is taken as if the I/O buffer were outside the map. These are all classified as program errors. If the device is temporarily under exclusive use, the node is stripped (placed in the TCT slough queue) and the next node is processed. A check must be made for the possible loss of the node. In this case, the symbiont goes back to the call to SS$SIO to process a new node.

At this point, the housekeeping operations are complete and the actual function of processing the data may begin. SS$GTW is called to retrieve data words from the I/O buffer. Since the symbiont has chosen to respond only to a WRITE request, we know an I/O buffer exists and the buffer contains valid data. All data will be processed and no special terminators will be recognized. Each time the compression routine is called, it will process one word, passed by SS$GTW. On return from SS$GTW, a check must be made for loss of the node. Since the processing of each word of data is done with the taskmaster "unlocked", this check must always be made. Since the symbiont supports the offline feature, this must be checked also. The response to these conditions is always the same and is described above. The only other special condition to check from SS$GTW is the buffer end condition. If this is true, there is no data returned and the shutdown of this operation occurs. This consists of the incrementing of the file position index and unqueuing the node. If buffer end is not indicated, a loop back to SS$GTW occurs until buffer end is detected. Of course, this loop will also end if the node gets pulled out or the device is put offline.

COMPRS also provides an entry to cause the symbiont's local buffer to be dumped to the designated real device. The ability to mark file boundaries is also provided. These features are provided simply by responding to the WEOF and HOME requests. The symbiont calls the particular entry point to COMPRS and then goes to the shutdown process.

Since the exclusive use feature is supported by this symbiont, the capability to TAKE and GIVE must be provided. On responding to those requests, the symbiont simply calls the appropriate routine (SS$TAK, SS$GIV) and proceeds to shutdown as for all other requests. An example Symbiont Flowchart is shown in Figure 3-1.

EXAMPLE:

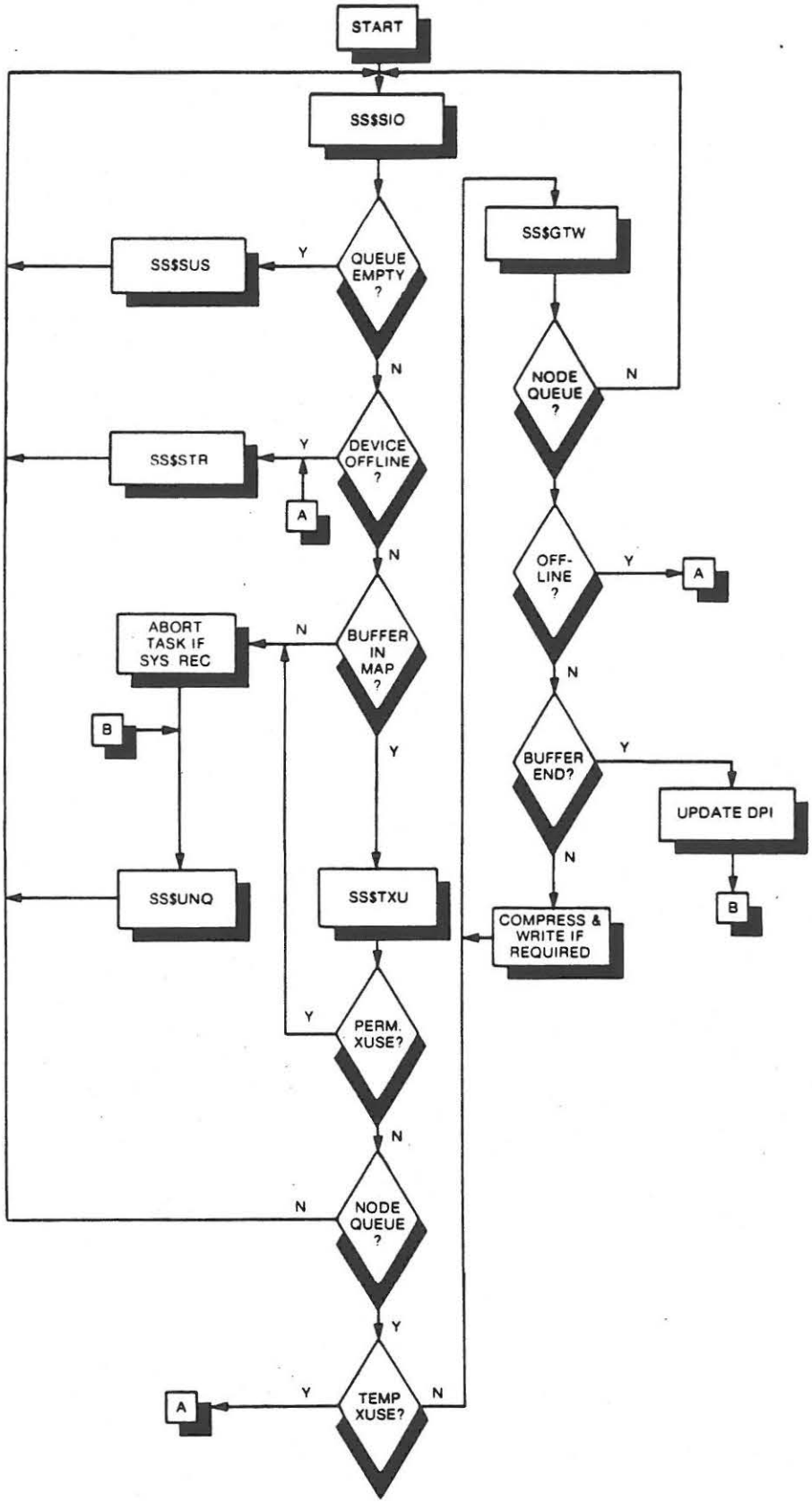


Figure 3-1. Symbiont Flowchart

3.8 EXAMPLE SYMBIONT CODING

```
MODCOMP CLASSIC MACRO ASSEMBLY E.3   02/15/83    PAGE 1
**** EXAMPLE MAX IV SYMBIONT CODING ****

                        PGM         COMPRS
                *       THIS PROGRAMS ACCEPTS DATA FROM THE CALLING TASK'S
                *       I/O BUFFER AND COMPRESSES IT INTO A LOCAL BUFFER.
                *       WHEN THE BUFFER IS FULL, IT WILL BE WRITTEN TO A
                *       REAL DEVICE.
                *
                *       THIS SYMBIONT RESPONDS ONLY TO THE WRITE, WEOF,
                *       AND HOME COMMANDS.  THE BODY OF ROUTINES CWRITE,
                *       CWEOF, AND CHOME ARE NOT INCLUDED IN THIS EXAMPLE,
                *       ONLY THE ENTRY AND RETURN POINTS.  THE READER
                *       SHOULD CHECK THE COMMENTS NEAR THE ENTRY POINTS
                *       TO SEE WHAT THE ROUTINES SHOULD DO.
                *
                *       NOTE THAT COUNTERS HAVE BEEN ADDED TO THIS CODE.
                *       SINCE MORE THAN ONE COUNTER EXISTS IN THIS MODULE,
                *       M4EDIT MUST BE USED FOR LINK EDITING THE PROGRAM.
                *
                         INS*       MC,IVUEQU
                         EXT        SS$SIO
                         EXT        SS$TXU
                         EXT        SS$SUS
                         EXT        SS$STR
                         EXT        SS$UNQ
                         EXT        SS$GTW
                         EXT        SS$KIL
                         EXT        SS$NOD
                         EXT        SS$GPI
                         EXT        SS$PPI
                         EXT        SS$TAK
                         EXT        SS$GIV
                *
                *
     2 0000              CTR        2
     2 0000 START        EQU        $           SYMBIONT ENTRY POINT
0000   E780              BLM,R8     SS$SUS      SUSPEND IF EMPTY QUEUE
0001 X 0003
     2 0002 COM010       EQU        $
0002   E780              BLM,R8     SS$SIO      INITIALIZE THE PDT
0003 X 0001
0004   A804              HNS,COM020             BRANCH IF READY TO PROCESS
0005   0E73              LCCC,R7                GET CC (TO USE AS INDEX)
0006   E70F              BRU*,R7    TAB200      BRANCH FOR PROPER
                *                                PROCESSING
0007 3 0000
     2 0008 COM020       EQU        $
0008   6D19              TRR,R1,R9              SAVE REX CALL NUMBER
0009   E780              BLM,R8     SS$TXU      TEST DEVICE EXCLUSIVE USE
000A X 0002
000B   A804              HNS,COM030             BRANCH IF EXCLUSIVE USE
000C   0E73              LCCC,R7                GET CC (TO USE AS INDEX)
000D   E70F              BRU*,R7    TAB210      BRANCH FOR PROPER PROCESSING
000E   0003
```

```
     2 000F COM030 EQU          $
000F   E709        BRU*,R1      TAB220      BRANCH FOR PROPER PROCESSING
0010 3 0006
     2 0011 COM040 EQU          $
0011   E780        BLM,R8       SS$GTW       GET A DATA WORD FROM BUFFER
0012 X 0006
0013   A804        HNS,COM050                TRANSFER IF DATA OK
0014   0E73        LCCC,R7                   GET CC (TO USE AS INDEX)
0015   E70F        BRU*,R7      TAB230       BRANCH FOR PROPER PROCESSING
0016 3 0011
     2 0017 COM050 EQU          $
0017   E780        BLM,R8       CWRITE       COMPRESS DATA IN LOCAL
               *                               BUFFER
0018 2 0046
0019   F778        HOP,COM040                REPEAT AS REQUIRED
               *
               *    PROCESS NODE STRIP CONDITION
               *
     2 001A COM060 EQU          $
001A   E780        BLM,R8       SS$STR       STRIP NODE FROM PDT
001B X 0004
001C   F764        HOP,START                 START OVER FOR NEXT NODE
               *
               *    PROCESS ABORTABLE CONDITIONS
               *
     2 001D COM070 EQU          $
001D   E780        BLM,R8       SS$NOD       GET INFORMATION
001E X 0008
001F   A802        HNS,COM080                HOP IF DATA RETURNED
0020   ABE2        HCR,COM010                TRANSFER IF NO DATA RETURNED
     2 0021 COM080 EQU          $
0021   6590        LBR,R9,B0                 SET ERROR FOR RETURN UFTSTA
0022   6CAA        ZRR,R10
0023   6630        TBR,R3,B0                 IS SYSTEM ERROR RECOVERY
               *                               REQUESTED
0024   AB93        HCR,COM130                TRANSFER IF LOCAL ERROR
               *                               RECOVERY
0025   E780        BLM,R8       SS$KIL       GO ABORT USER TASK
0026 X 0007
0027   A810        HNS,COM130                HOP TO UNQUEUE NODE
0028   ABDA        HCR,COM101                TRANSFER IF NODE NOT PRESENT
0029   F70E        HOP,COM130
     2 002A COM090 EQU          $
002A   E780        BLM,R8       SS$GPI       GET DEVICE POSITION INDEX
002B X 0009
002C   A802        HNS,COM100                TRANSFER IF NODE PRESENT
002D   ABD5        HCR,COM010                TRANSFER IF NODE GONE
     2 002E COM100 EQU          $
002E   607F        ABR,R7,B15                INCREMENT 32-BIT DPI
002F   AB82        HCR,COM110                HOP IF CARRY INTO MOST
               *                               SIGNIFICANT
```

```
0030   606F          ABR,R6,B15              INCREMENT MOST SIGNIFICANT
                     *                         HALF
     2 0031 COM110 EQU          $
0031   E780          BLM,R8      SS$PPI      PUT NEW DPI IN NODE
0032 X 000A
0033   A802          HNS,COM120              TRANSFER IF NODE PRESENT
0034   ABCE          HCR,COM010              TRANSFER IF NODE NOT PRESENT
     2 0035 COM120 EQU          $
0035   6C99          ZRR,R9
0036   6CAA          ZRR,R10
     2 0037 COM130 EQU          $
0037   E780          BLM,R8      SS$UNQ      UNQUEUE I/O NODE
0038 X 0005
0039   F749          HOP,COM010              SEE IF MORE NODES TO PROCESS
     2 003A COM140 EQU          $
003A   E780          BLM,R8       CWEOF      WRITE EOF TO REAL DEVICE
003B 2 0048
003C   F76E          HOP,COM090              UPDATE DPI
     2 003D COM150 EQU           $
003D   E780          BLM,R8       CHOME      HOME REAL DEVICE
003E 2 0047
003F   F76B          HOP,COM090              UDPATE DPI
     2 0040 COM160 EQU           $
0040   E780          BLM,R8       SS$TAK     TAKE EXCLUSIVE USE
0041 X 000B
0042   F773          HOP,COM120
     2 0043 COM170 EQU           $
0043   E780          BLM,R8       SS$GIV     GIVE UP EXCLUSIVE USE
0044 X 000C
0045   F770          HOP,COM120
                     *
                     *       CWRITE WILL COMPRESS THE DATA ONE WORD AT A TIME,
                     *       EACH TIME IT IS CALLED.
                     *
     2 0046 CWRITE EQU           $
                     *
                     *       CODE FOR CWRITE SHOULD BE ENTERED HERE
                     *
0046   FF08          BRX,R8                  RETURN
                     *
                     *
                     *
                     *       CHOME WILL FORCE THE COMPRESSED BUFFER TO BE
                     *       WRITTEN EVEN IF NOT COMPLETELY FULL.
                     *
     2 0047 CHOME  EQU           $
                     *
                     *       CODE FOR CHOME SHOULD BE ENTERED HERE
                     *
```

```
0047   FF08          BRX,R8                    RETURN
                *
                *
                *
                *       CWEOF WILL DO A HOME, THEN WRITE AN END OF FILE
                *       TO THE REAL DEVICE.
                *
     2  0048 CWEOF   EQU          $
                *
                *       CODE FOR CWEOF SHOULD BE ENTERED HERE
                *
0048   FF08          BRX,R8                    RETURN
                *
                *
                *       THE FOLLOWING TABLES ARE USED TO PROCESS
                *       CONDITIONS NOTED IN PROCESSING.
                *
     3  0000          CTR          3
     3  0000 TAB200  EQU          $
0000 2  001A          DFC          COM060     DEVICE OFFLINE
0001 2  0000          DFC          START      QUEUE EMPTY
0002 2  001D          DFC          COM070     ILLEGAL BUFFER
                *
     3  0003 TAB210  EQU          $
0003 2  0002          DFC          COM101     NODE REMOVED
0004 2  001A          DFC          COM060     TEMPORARY EXCLUSIVE USE
0005 2  001D          DFC          COM070     PERMANENT EXCLUSIVE USE
     3  0006 TAB220  EQU          $
0006 2  0035          DFC          COM120     ( 0) READ - DO NOTHING
0007 2  0011          DFC          COM040     ( 1) WRITE - COMPRESS DATA
0008 2  0035          DFC          COM120     ( 2) REWIND - DO NOTHING
0009 2  0035          DFC          COM120     ( 3) BKF - DO NOTHING
000A 2  0035          DFC          COM120     ( 4) BKR - DO NOTHING
000B 2  0035          DFC          COM120     ( 5) AVR - DO NOTHING
000C 2  0035          DFC          COM120     ( 6) AVF - DO NOTHING
000D 2  003A          DFC          COM140     ( 7) WEOF - DUMP BUFFER,
                *                                    WRITE EOF
000E 2  003D          DFC          COM150     ( 8) HOME - DUMP BUFFER
000F 2  0040          DFC          COM160     ( 9) TAKE EXCLUSIVE USE
0010 2  0043          DFC          COM170     (10) GIVE EXCLUSIVE USE
                *
     3  0011 TAB230  EQU          $
0011 2  0002          DFC          COM010     NODE REMOVED
0012 2  001A          DFC          COM060     DEVICE OFFLINE
0013 2  002A          DFC          COM090     END OF BUFFER
                *
0014 2  0000          END          START
```

3.9 SKELETON SYMBIONT

The purpose of the symbiont interface program is to enhance the implementation of symbionts to the MAX IV operating system. This source.program is the body of a resident or prescheduled task that requires additional user source to complete. The completed program may be added to the system as a handler simulator without modifying the executive.

Control is transferred from the calling task to one of 11 entry points within the symbiont, depending upon the REX service called. The symbiont user may supply up to 11 subroutines to allow the symbiont task to process I/O REX calls. A subroutine may return to either of three defined addresses. The subroutines are used to move the data within the symbiont buffer to or from the special I/O device. REX calls may be executed from within any subroutine.

A symbiont data initialization subroutine may also be supplied by the user to initialize data bases.

If the symbiont to be implemented requires interfacing with the requesting I/O operation during I/O task-level or during test assignment requests then the symbiont routine suffixes can be included on the SYMCONTROL statement during SYSGEN.

3.10 IMPLEMENTATION

3.10.1 FIRST TIME INITIALIZATION

The symbiont task initializes itself by branching to the user supplied initialization subroutine (UINIT.). All registers, except R8 which holds the return link, are volatile. This subroutine is only used at symbiont task activation.

3.10.2 USER SUPPLIED SUBROUTINES

The symbiont user must supply the following subroutines:

READ
WRITD
REWD
BKFD
BKRD
AVRD
AVFD
WEOFD
HOMED
TEXUD
GEXUD

The following arguments are passed to each subroutine:

R1 = REX service number for I/O node being processed
R2 = Node type and status (NODSTA)
R3 = UFT option word (UFTOPT)
R4 = Address of calling task buffer in task Ø-space (NODBUF)
R5 = Map image/actual Ø-space page where calling task buffer resides (NODMAP)
R6 = Map image length (NODLEN)
R7 = Transfer count in bytes (NODCNT)

The information in registers R1,R2,..,R7 for the I/O node being processed at the time are also stored in table I$NOD within the symbiont body. All registers are volatile.

Before branching to a user subroutine, the symbiont will set to zero the word at label S$UFTSTA within its body. Upon return from the user supplied routine, the contents of that word must be set to the final status of the I/O operation, if needed, so that the symbiont can transfer it to word (UFTSTA) in the UFT of the calling task when the I/O completes.

In case of a READ operation, if the byte count transferred to the symbiont internal buffer (B$BUF) is less than the byte count specified in the calling task UFT (NODCNT), the user must store that byte count at label S$BCT before returning from his routine.

The user supplied routine must return to label RTDPI. (BRU RTDPI.), if the Device Position Index (UFTDPI) in the calling task UFT needs to be incremented by one or to label RETRN. (BRU RETRN.) if not.

If the user wishes that the symbiont EXIT after the processing of all nodes in its queue (normal shutdown), Bit 15 must be set to one in the symbiont flag word S$FLAG before returning from the user supplied routine.

A user supplied routine could force the calling task that has queued the I/O node to abort by placing any three character abort code (CAN-code) in R13 and branching to label K$ILL in the symbiont. However, if the option word in the UFT of the calling task does not indicate System Recovery (Bit Ø=1 in UFTOPT), the task will not abort and only the node will be unqueued.

3.1Ø.3 SYMBIONT ASSEMBLY AND LINK-EDITING

The symbiont is assembled using the CLASSIC Assembler (M5ASM or M5A). The Macro-files IVMAC and IVUEQU that the symbiont references are inserted from file MC.

After assembly, the symbiont should be link-edited with file UL assigned to AL4 (UL=AL4) in order to load the symbiont external subroutines.

3.10.4 SYMBIONT CATALOGING

The symbiont is normally cataloged on a directoried load module File such as the LM file by means of the Task Overlay Cataloger (TOC).

The following is a sample job stream for cataloging a symbiont task whose name is XYP:

```
$JOB
$ASS       BI=(medium where link-edited symbiont resides)
$EXE       TOC
FILE       LM
TASK       XYP,2
PECULIAR   SYMBIONT,PRIVILEGED
ID         XYPLOTTER
LOGFILE    F1,DSA
CATALOG
```

In the above example, file F1 is used internally by the symbiont task.

3.11 SYMBIONT SYSTEM GENERATION

The following SYSGEN statements are needed for symbiont tasks:

3.11.1 SYMCONTROLLER

```
                    1     2          3          4
SYMCONTROLLER   name[,buffer][,TLsuffix][,TAsuffix]
```

name - Parameter 1 is the name of the symbiont task.

buffer - Parameter 2 is the number of words to be configured in the symbiont PDT optional buffer.

TLsuffix - Parameter 3 is a 1- to 3-character CAN-codable suffix to be added to the prefix TL$ to be used as an external reference for the symbiont optional I/O task-level routine.

TAsuffix - Parameter 4 is a 1- to 3-character CAN-codable suffix to be added to the prefix TA$ to be used as an external reference for the symbiont optional test assign routine.

3.11.2 DEVICE

```
                    1    2   3    4
     DEVICE         pdev,name,bpr,option...
```

pdev - Parameter 1 is the pseudo-device name on which user tasks will perform I/O operation.

name - Parameter 2 is the symbiont task name.

bpr - Parameter 3 is the record length in bytes.

option - Parameter 4 is the device characteristics/option.

3.11.3 PRESCHEDULE

```
                    1    2     3
     PRESCHEDULE    name,level,file
```

name - Parameter 1 is the symbiont task name.

level - Parameter 2 is the priority level for the symbiont task.

file - Parameter 3 is the global file name from which the symbiont will be loaded.

In general a symbiont task always executes at a higher priority level than the calling task(s).

Please refer to the MAX IV SYSGEN, System Guide Manual, for more information on SYMCONTROLLER, DEVICE, PRESCHEDULE, and Global file assignment in case user's tasks wish to perform I/O to a Global file that is assigned to the symbiont task pseudo-device.

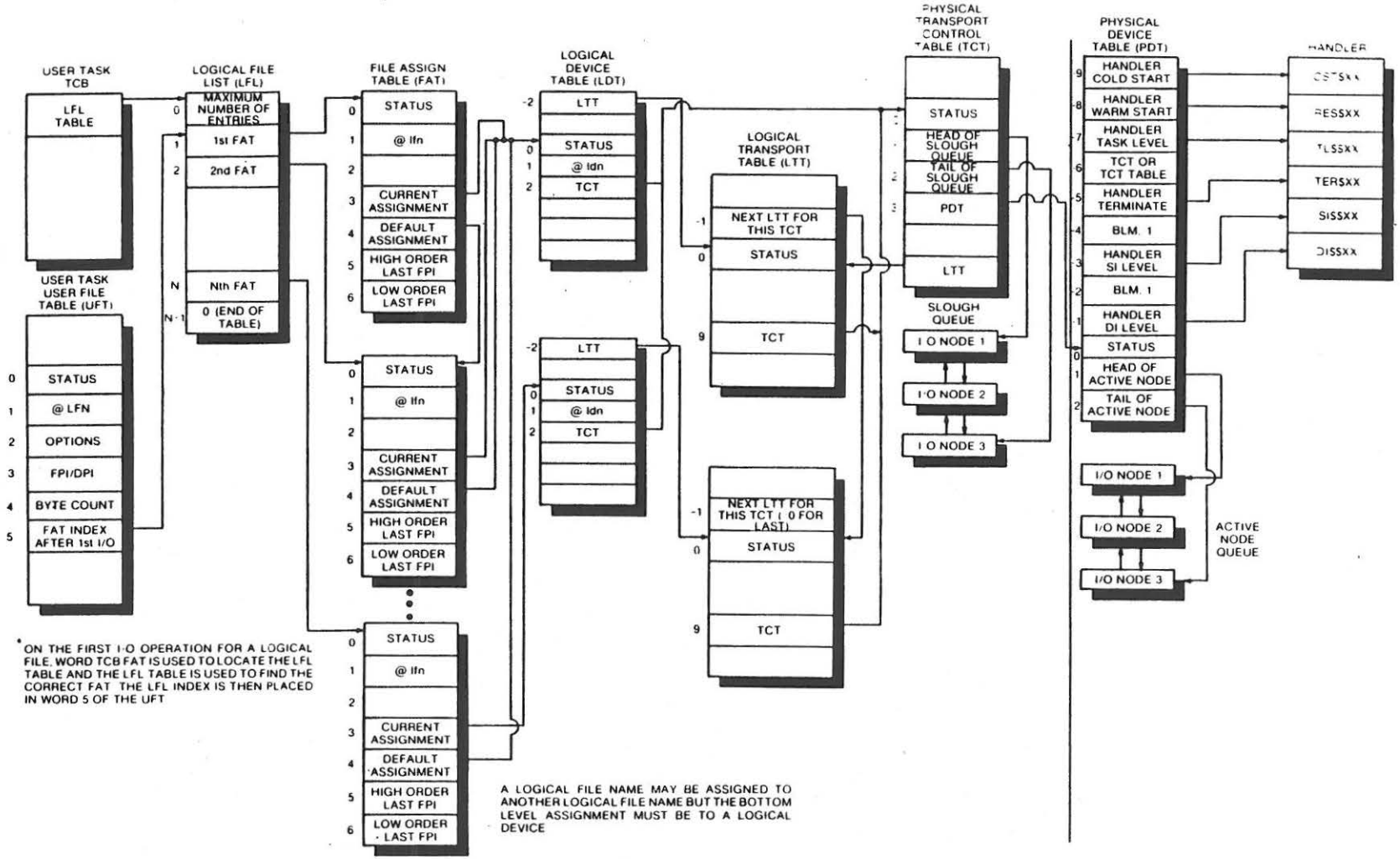
3.12 MAX IV SKELETON SYMBIONT

```
         TTL       MAX IV SKELETON SYMBIONT                          *D
         PGM       S.SKEL                                            *D
         INS       MC,IVMAC
         INS*      MC,IVUEQU
         INT       C$NAME,T$NAME,E$NAME
         EXT       SS$SUS,SS$SIO,SS$TXU,SS$STR,SS$UNQ,SS$KIL,SS$NOD
         EXT       SS$GPI,SS$PPI,SS$GBF,SS$SBF,SS$EXIT,SS$EXF
         SPC
*        NOTE: THE SYMBIONT INTERFACE WILL DO A  'BLM,R8  UINIT.' FOR
*              FIRST TIME INITIALIZATION WITH ALL REGISTERS
*              INDETERMINATE. ALL REGISTERS MAY BE DESTROYED
*              UINIT.  WILL ONLY BE CALLED ONE TIME AT START-UP
*              THE USER SHOULD RETURN FROM  UINIT.  WITH A BRX,R8
*              OR A 'BRU   S$SUS'  INSTRUCTION.
*
*        NOTE:  THE USER MUST SUPPLY A BUFFER WHOSE ADDRESS IS AT B$BUF
*                 AND WHOSE LENGTH IS EQUAL OR LESS THAN THE
*                 MAXIMUM RECORD LENGTH USED BY THE CALLER (SPECIFIED
*                 AT SYSGEN TIME ON THE "DEVICE" STATEMENT)
         SPC
*            ON ALL WRITES, THE CALLER'S BUFFER WILL BE MOVED TO B$BUF
         SPC
*            ON ALL READS, CONTENT OF B$BUF WILL BE MOVED TO CALLER
*              BUFFER
*        NOTE: THE FOLLOWING REGISTERS ARE SET BEFORE CALLING THE USER
*              SUPPLIED ROUTINES (READ.),(WRITD.),(REWD.),(BKFD.),
*              (BKRD.),(AVRD.),(AVFD.),(WEOFD.),(HOMED.),(TEXUD.),
*              (GEXUD.)
         SPC
*        (R1) = REX NUMBER OF CALL
*        (R2) = NODE TYPE AND STATUS (NODSTA)
*        (R3) = UFT OPTION WORD (UFTOPT)
*        (R4) = ADDRESS OF USER BUFFER IN HIS TASK SPACE (NODBUF)
*        (R5) = MAP IMAGE ACTUAL PAGE WHERE BUFFER RESIDES  (NODMAP)
*        (R6) = MAP IMAGE LENGTH  (NODLEN)
*        (R7) = TRANSFER BYTE COUNT (NODCNT)
         SPC
*              ABOVE NODE INFORMATION IS ALSO STORED IN TABLE I$NOD
*              BEFORE PROCESSING THE USER SUPPLIED ROUTINES.
*
*        NOTE:  ALL OF THE ABOVE REGISTERS MAY BE DESTROYED BY THE USER
*               PART OF THE SYMBIONT
*
*
*        NOTE:  THE USER MUST  RETURN TO LABEL RTDPI, IF HE WISHES TO
*               UPDATE UFTDPI IN UFT OF CALLING TASK.
         SPC
*                IF THE USER WISHES ONLY TO RESTORE  UFTSTA
*           AND UFTBCT TO THE SYMBIONT INTERNAL VALUES AT S$UFTSTA
*           AND S$BCT AND UNQUEUE NODE,  HE MUST RETURN TO LABEL RETRN.
*
*        NOTE :   BEFORE RETURNING AFTER PROCESSING A NODE (I.E. BEFORE
*                 A BRU RETRN...) USER MUST STORE ANY ERROR STATUS THAT
*                 SHOULD BE RELAYED TO WORD UFTSTA IN USER'S TASK UFT
*                 IN WORD S$UFTSTA; OTHERWISE, (UFTSTA)=Ø UPON
```

```
*               UPON COMPLETION OF I/O OPERATION.
      SPC
*               ALSO, IF BYTE COUNT NEED TO BE RELAYED TO WORD UFTBCT
*               IN USER'S TASK UFT, THE SYMBIONT MUST STORE THIS
*               VALUE IN WORD S$BCT;OTHERWISE, (UFTBCT)=Ø UPON
*               COMPLETION OF I/O.
      SPC
*               IF THE USER WISHES THAT THE SYMBIONT EXITS AFTER
*               PROCESSING ALL NODES IN ITS QUEUE (NORMAL SHUTDOWN), HE
*               MUST SET BIT 15=1 IN SYMBIONT FLAG WORD S$FLAG BEFORE
*               RETURNING TO RTDPI.  OR RETRN.
      SPC
*     NOTE :    DURING THE PROCESSING OF A USER ROUTINE, THE CALLING
*               TASK COULD BE CAUSED TO ABORT AND ALSO THE SYMBIONT TO
*               EXIT BY PLACING A THREE CHARACTER 'CANNED' ABORT CODE
*               IN REGISTER 13 AND BRANCHING TO LABEL K$ILL... THIS IS
*               PROVIDED ONLY IF THE NODE BEING PROCESSED INDICATES
*               THAT SYSTEM ERROR RECOVERY IS IN EFFECT.
      EJT
C$NAME EQU          $               START OF COMMON (USER'S SUPPLIED)
       SPC 1
T$NAME EQU          $               TRANSFER ADDRESS (USER'S SUPPLIED)
       BLM,R8       UINIT.          GO PROCESS USER'S INITIALIZATION
R$XIT  TBMM,B15     S$FLAG          TEST SYMBIONT FLAG WORD FOR EXIT
       HCR,S$SUS                    HOP IF NO EXIT
       BLM,R8       SS$EXIT         EXIT FROM SYMBIONT IF NO NODES
       ZBMM,B15     S$FLAG          RESET EXIT SYMBIONT FLAG BIT
S$SUS  BLM,R8       SS$SUS          SUSPEND SYMBIONT IF QUEUE EMPTY
S$TART BLM,R8       SS$SIO          INITIALIZE PDT OF SYMBIONT
       HCS,R$XIT                    HOP IF NO NODES IN QUEUE
       HOS,I$BUF                    HOP IF ILLEGAL BUFFER
       HNR,Q$STR                    HOP IF DEVICE OFFLINE
       TRR,R1,R9                    (R1) = REX SERVICE NUMBER
       BLM,R8       SS$TXU          GO CHECK FOR EXCLUSIVE USE
       HCS,Q$STR    HOP TO STRIP NODE  IF TEMPORARY EXCL. USE
       HOS,X$LIM    HOP/KILL USR TSK IF PERM. EXCL. USE/LIM. PR. ERROR
       HNR,S$TART                   HOP  IF NODE REMOVED FROM QUEUE
       SBRB,R1,ONE  G$NOD           BRU IF NOT WRITE OPERATION
       LDI,R14      B$BUF           NR14)= ADDR TO MOVE USER BUFFER IN
       BLM,R8       SS$GBF          MOVE CONTENT OF USR BUFFER TO B$BUF
       HNS,G$NOD                    HOP, IF BUFFER MOVED O.K.
H$HOP  HCS,Q$STR                    GO STRIP NODE IF DEVICE OFFLINE
H$HIP  HOP,S$TART                   HOP BACK IF NODE WAS REMOVED
       SPC
G$NOD  BLM,R8       SS$NOD          GET INFORMATION ABOUT NODE
       TRR,R1,R2                    (R1)=(R2)=NODSTA
       ETI,R1       #FFFØ           (R1)=REX NUMBER OF CALL
       ZRR,R15                      (R15) = Ø
       STM,R15      S$UFTSTA        ZERO SYMBIONT INTERLA UFTSTA WORD
       SFM,R1       I$NOD           SAVE REGISTERS  R1,R2,...,R7
       BRU*,R1      T$TBL           GO PROCESS AT APPROPRIATE ENTRY
RTDPI. EQU          $               UPDATE DEV POSITION INDEX (UFTDPI)
       BLM,R8       SS$GPI          GET DEVICE POSITION INDEX (UFTDPI)
       HNS,$+2                      HOP IF FPI VALID
       HOP,H$HOP                    GO CHECK FOR PROBLEM
       ABR,R7,ONE                   INCREMENET 3I BIT DPI BY ONE
       HCR,$+2
```

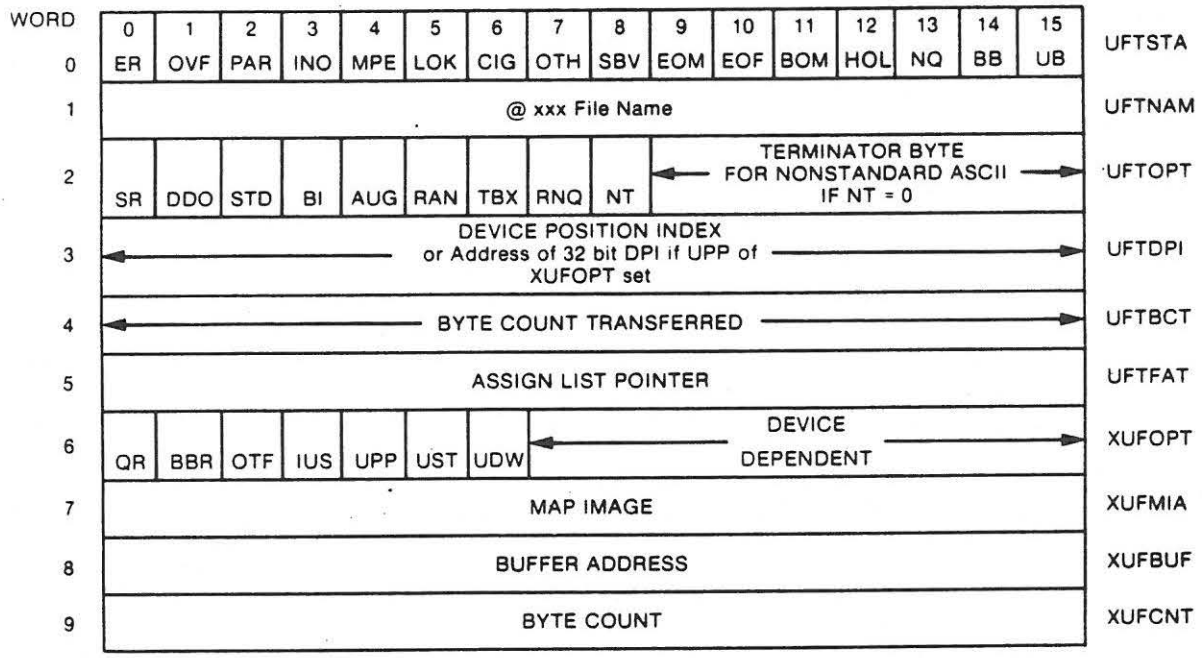
```
         ABR,R6,ONE
         BLM,R8        SS$PPI          STORE NEW DPI BACK IN UFT
         HNR,H$HOP                     GO CHECK PROBLEM IF DPI NOT UPDATED
RETRN.   EQU           $               RETURN AND DO NOT UPDATE UFTDPI
         TBMM,B1       I$NOD           TEST IF NODE IS FOR READ I/O
         HZR,$+7                       HOP, IF NOT READ
         LDI,R14       B$BUF           (R14)= SYMBIONT BUFFER
         BLM,R8        SS$SBF          GO MOVE IT TO USER'S TASK BUFFER
         HNS,$+2                       HOP,  IF BUFFER MOVED O. K.
         HOP,H$HOP                     GO CHECK FOR PROBLEM
         LDM,9         S$UFTSTA        (R9)=(UFTSTA)
         LDM,10        S$BCT           (R10)=(UFTBCT)
Q$UNQ    BLM,R8        SS$UNQ          UNQUEUE NODE: (R9)=UFTSTA,
*                                        (R10)=UFTBCT
         HOP,S$START                   RETURN TO PROCESS NEW NODES, IF ANY
         SPC
*                       STRIP NODE FROM QUEUE
         SPC
Q$STR    BLM,R8        SS$STR          GO STRIP NODE
         HOP,S$START                   GO BACK TO START
         SPC
I$BUF    LDI,R1        @IBF            (R1)=ABORT CODE FOR ILLEGAL BUFFER
         HOP,K$ILL                     GO KILL USER TASK
X$LIM    LDI,R1        @XLM            (R1)=ABORT CODE
K$ILL    BLM,R8        SS$NOD          GET INFORMATION ABOUT NODE
         HNS,$+2                       HOP IF NODE STILL IN QUEUE
         HCR,H$HIP                     HOP IF NODE WAS REMOVED FROM QUEUE
         LDI,R9        #8100           (R9)=(UFTSTA) IN CASE SYST RECOVERY
         ZRR,10                        (R10) = UFTBTC
         TBR,R3,SR                     TEST FOR SYSTEM RECOVERY IN UFTSTA
         HCR,Q$UNQ                     HOP IF USER ERROR RECOVERY
         TRR,R13,R1                    (R13) = ABORT CODE
         BLM,R8        SS$KIL          ABORT USER TASK
         BLM,R8        SS$UNQ          UNQUEUE NODE
S$XIT    BRU           SS$EXF          FORCE EXIT FROM SYMBIONT
         SPC 1
UINIT.   BRX,R8                        USER'S INITIALIZATION ROUTINE
         SPC
*                       U S E R   C O D E D   S U B R O U T I N E S
         SPC
T$TBL    DFC           READ.           0   USER SUPPLIED READ ROUTINE
         DFC           WRITD.          1   USER SUPPLIED WRITE ROUTINE
         DFC           REWD.           2   USER SUPPLIED REWIND ROUTINE
         DFC           BKFD.           3   USER SUPPLIED BKFILE ROUTINE
         DFC           BKRD.           4   USER SUPPLIED BKRECORD ROUTINE
         DFC           AVRD.           5   USER SUPPLIED AVRECORD ROUTINE
         DFC           AVFD.           6   USER SUPPLIED AVFILE ROUTINE
         DFC           WEOFD.          7   USER SUPPLIED WEOF ROUTINE
         DFC           HOMED.          8   USER SUPPLIED HOME ROUTINE
         DFC           TEXUD.          9   TAKE EXCLUSIVE USE
         DFC           GEXUD.          #A  GIVE-UP EXCLUSIVE USE
         SPC
S$UFTSTA HLT           UFTSTA TO BE PLACED IN USER'S UFT
S$FLAG   HLT           SYMBIONT FLAG WORD (BIT 15=1, CAUSES SYMBIONT TO
*                        EXIT
I$NOD    DFC  0        REX CALL NUMBER
         DFC  0        NODSTA
```

```
        DFC   Ø         UFTOPT
        DFC   Ø         NODBUF
        DFC   Ø         NODMAP
        DFC   Ø         NODLEN
S$BCT   DFC   Ø         NODCNT
        SPC
B$BUF   RES   128,Ø     SYMBIONT BUFFER (128 WORDS LENGTH IN THIS EXAMPLE)
        SPC         3                                                    *D
READ.   EQU         $                                                    *D
WRITD.  EQU         $                                                    *D
REWD.   EQU         $                                                    *D
BKFD.   EQU         $                                                    *D
BKRD.   EQU         $                                                    *D
AVRD.   EQU         $                                                    *D
AVFD.   EQU         $                                                    *D
WEOFD.  EQU         $                                                    *D
HOMED.  EQU         $                                                    *D
TEXUD.  EQU         $                                                    *D
GEXUD.  EQU         $                                                    *D
      SPC
```


USER TASK TCB
LFL TABLE
USER TASK USER FILE TABLE (UFT)
0 STATUS
1 @ LFN
2 OPTIONS
3 FPI/DPI
4 BYTE COUNT
5 FAT INDEX AFTER 1st I/O
LOGICAL FILE LIST (LFL)
0 MAXIMUM NUMBER OF ENTRIES
1 1st FAT
2 2nd FAT
N Nth FAT
N+1 0 (END OF TABLE)
FILE ASSIGN TABLE (FAT)
0 STATUS
1 @ lfn
2
3 CURRENT ASSIGNMENT
4 DEFAULT ASSIGNMENT
5 HIGH ORDER LAST FPI
6 LOW ORDER LAST FPI
LOGICAL DEVICE TABLE (LDT)
-2 LTT
0 STATUS
1 @ ldn
2 TCT
LOGICAL TRANSPORT TABLE (LTT)
-1 NEXT LTT FOR THIS TCT
0 STATUS
9 TCT
-1 NEXT LTT FOR THIS TCT (0 FOR LAST)
PHYSICAL TRANSPORT CONTROL TABLE (TCT)
STATUS
HEAD OF SLOUGH QUEUE
TAIL OF SLOUGH QUEUE
PDT
LTT
SLOUGH QUEUE
I O NODE 1
I O NODE 2
I O NODE 3
PHYSICAL DEVICE TABLE (PDT)
9 HANDLER COLD START
8 HANDLER WARM START
7 HANDLER TASK LEVEL
6 TCT OR TCT TABLE
5 HANDLER TERMINATE
4 BLM. 1
3 HANDLER SI LEVEL
2 BLM. 1
1 HANDLER DI LEVEL
0 STATUS
1 HEAD OF ACTIVE NODE
2 TAIL OF ACTIVE NODE
I/O NODE 1
I/O NODE 2
I/O NODE 3
ACTIVE NODE QUEUE
HANDLER
CSTSXX
RESSXX
TLSSXX
TERSXX
SISSXX
DISSXX
ON THE FIRST I-O OPERATION FOR A LOGICAL FILE, WORD TCB FAT IS USED TO LOCATE THE LFL TABLE AND THE LFL TABLE IS USED TO FIND THE CORRECT FAT. THE LFL INDEX IS THEN PLACED IN WORD 5 OF THE UFT
A LOGICAL FILE NAME MAY BE ASSIGNED TO ANOTHER LOGICAL FILE NAME BUT THE BOTTOM LEVEL ASSIGNMENT MUST BE TO A LOGICAL DEVICE

APPENDIX B
MAX IV UFT

MAX IV UFT



Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 3624 FT. LAUDERDALE, FL 33309

POSTAGE WILL BE PAID BY ADDRESSEE

MODULAR COMPUTER SYSTEMS
1650 W. McNAB ROAD
P.O. BOX 6099
FT. LAUDERDALE, FLORIDA 33310

Attention: TECHNICAL PUBLICATIONS, M.S. #85

Fold

MODCOMP®

Please comment on the publication's completeness, accuracy, and readability. We also appreciate any general suggestions you may have to improve this publication.

If you found errors in this publication, please specify the page number or include a copy of the page with your remarks.

Your comments will be promptly investigated and appropriate action will be taken. If you require a written answer, please check the box and include your address below.

☐

Comments: _____

Manual Title _____

Manual Order Number _____ Issue Date _____

Name _____ Position _____

Company _____

Address _____

_____ Telephone ( ) _____

MCS 148D (1/83)

Corporate Headquarters:
MODULAR COMPUTER SYSTEMS, Inc., 1650 West McNab Road, P.O. Box 6099, Ft. Lauderdale, FL 33310, Tel: (305) 974-1380, TWX: 310-372-7837

International Headquarters:
MODULAR COMPUTER SERVICES, Inc., The Business Centre, Molly Millars Lane, Wokingham, Berkshire, RG11 2JQ, UK, Tel: 0734-786808, TLX: 851849149

Latin American Sales Headquarters:
MODULAR COMPUTER SYSTEMS, Inc., 1650 West McNab Road, P.O. Box 6099, Ft. Lauderdale, FL 33310, Tel: (305) 975-6562, TLX: 3727852

Canadian Headquarters:
MODCOMP Canada, Ltd., 400 Matheson Blvd. East, Unit 24, Mississauga, Ontario, Canada L4Z 1N8, Tel: (416) 890-0666, TELEX: 06-961279

SALES & SERVICE LOCATIONS THROUGHOUT THE WORLD

Printed in the U.S.A.